CodeWarrior Development Studio for Microcontrollers V10.x Targeting Manual

Document Number: CWMCUDBGUG
Rev 10.6, 03/2014

Contents

Section number **Title** **Page**

Chapter 1 Introduction

Chapter 2 Working with Projects

Chapter 3
Build Properties for Bareboard Projects

Section number Title Page

Section number Title Page

Section number Title Page

Chapter 4 Working with Debugger

Section number	Title	Page

Chapter 5
Kinetis Cache Viewer

Chapter 6 Multicore Debugging

Chapter 7 CodeWarrior Command Line Debugging

Chapter 8
Build Properties for Linux Project

Chapter 9
Connections - HCS08 Architecture

Chapter 10
Connections - RS08

Chapter 11
Connections - ColdFire V1/ColdFire+ V1

Chapter 12
Connections - ColdFire V2/3/4

Chapter 13
Connections - Qorivva MPC55xx/56xx

Chapter 14
Connections — Kinetis Architecture

Section number Title Page

Chapter 15
Connections - DSC Architecture

Chapter 16
Connections - S12Z Architecture

Chapter 17
Common Connection Features

Chapter 18
CRC Utility for All Architectures

Chapter 19
How to...

Chapter 20
S12Z IEEE-754 Floating Point Library

Chapter 1
Introduction

Introduces you to the information layout of the manual.

This manual explains how to use the CodeWarrior Development Studio for Microcontrollers V10.x product. This chapter presents an overview of the manual.

The topics in this chapter are:

- Release Notes - Lists new features, bug fixes, and incompatibilities
- About this Manual - Describes the contents of this manual
- Accompanying Documentation - Describes supplementary CodeWarrior documentation, third-party documentation, and references.

1.1 Release Notes

Before using the CodeWarrior IDE, read the developer notes. These notes contain important information about last-minute changes, bug fixes, incompatible elements, or other topics that may not be included in this manual.

NOTE

The release notes for specific components of the CodeWarrior IDE are located in the `Release_Notes` folder in the CodeWarrior installation directory.

1.2 About this Manual

Each chapter of this manual describes a different area of software development. The following table lists the contents of this manual.

Table 1-1. Manual Contents

Chapter / Appendix	Description
Introduction	This chapter.
Working with Projects	Explains how to use the CodeWarrior tools to create and work with projects.
Build Properties for Bareboard Projects	Explains build properties for Microcontrollers bareboard project.
Working with Debugger	Explains how to use the CodeWarrior development tools to debug a program executing on the simulator or microcontroller.
Kinetis Cache Viewer	Describes the Kinetis Cache Viewer available in CodeWarrior for Microcontrollers Version 10.x.
Multicore Debugging	Explains how to define multiple, arbitrary groupings of cores and perform multicore operations.
CodeWarrior Command Line Debugging	Explains how CodeWarrior supports the command-line interface.
Build Properties for Linux Project	Explains build properties for Microcontrollers Linux project.
Connections - HCS08 Architecture	Describes the features and settings of the connections that interface the CodeWarrior debugger with the HCS08-based bareboard target and allows it to debug program code on the target.
Connections - RS08	Describes the features and settings of the connections that interface the CodeWarrior debugger with the RS08-based bareboard target, and allow it to debug program code on the target.
Connections - ColdFire V1/ColdFire+ V1	Describes the features and settings of the connections that interface the CodeWarrior debugger with the ColdFire V1-based 10.x target, and allow it to debug program code on the target.
Connections - ColdFire V2/3/4	Describes the features and settings of the connections that interface the CodeWarrior debugger with the ColdFire V2/3/4-based bareboard target, and allow it to debug program code on the target.
Connections - Qorivva MPC55xx/56xx	Describes the features and settings of the connections that interface the CodeWarrior debugger with the Power Architecture bareboard target.
Connections — Kinetis Architecture	Describes the features and settings of the connections that interface the CodeWarrior debugger with the Kinetis-based bareboard target.
Connections - DSC Architecture	Describes the features and settings of the connections that interface the CodeWarrior debugger with the Digital Signal Controller (DSC) target board.
Connections - S12Z Architecture	Describes the features and settings of the connections that interface the CodeWarrior debugger with the S12Z target board.

Table continues on the next page...

Table 1-1. Manual Contents (continued)

Chapter / Appendix	Description
Common Connection Features	Explains how to use the CodeWarrior hardware tools for board bring-up, test, and analysis. Also, explains how to manipulate target memory.
CRC Utility for All Architectures	Describes the CRC utility for all architectures.
How to...	Describes additional how to tasks.
S12Z IEEE-754 Floating Point Library	Describes the implementation of the floating point arithmetic as described in the IEEE-754 standard. The following floating point routines for the S12Z device family are implemented

1.3 Accompanying Documentation

- To view the online help for the CodeWarrior tools, select **Help > Help Contents** from the IDE's menu bar. Next, select **Microcontrollers V10.x > Targeting Microcontrollers > Microcontrollers V10.x Targeting Manual** from the **Contents** list.
- For late-breaking information about new features, bug fixes, known problems, and incompatibilities, read the release notes in this folder:

  ```
  <CWInstallDir>\MCU\Release_Notes
  ```

  where *CWInstallDir* is the directory that CodeWarrior was installed into.
- For general information about the CodeWarrior IDE and debugger, refer to the **CodeWarrior Common Features Guide**in this folder:

  ```
  <CWInstallDir>\MCU\Help\PDF
  ```

Chapter 2
Working with Projects

This chapter explains how to use the CodeWarrior to create and work with projects.

NOTE

The example projects can be accessed from the Help > Welcome page and are located in [install_dir]/MCU/CodeWarrior_Examples, where [install_dir] is the location of the CodeWarrior layout . You can use the Eclipse Import Wizard to import example projects into your workspace.

The topics in this chapter are:

- Types of Projects
- New Bareboard Project Wizard
- New Linux/uClinux Application Project Wizard
- Creating Projects
- Building Projects
- Debugging Projects
- Deleting Projects
- Porting Classic DSC Project to Eclipse Project

2.1 Types of Projects

CodeWarrior projects organize files and various compiler, linker, and debugger settings associated with the applications or libraries you develop. You use Microcontrollers New Project Wizard to create new projects that group these files and settings into build and launch configurations. This section describes the different types of projects you can create:

- Bareboard Projects
- Linux Projects

2.1.1 Bareboard Projects

With CodeWarrior Development Studio for Microcontrollers, you can create a variety of projects that create ELF executable binary files that run directly on a given target simulator or board, without a Linux operating system. The type of project you create is based on selections you make in the **New Bareboard Project** wizard.

2.1.2 Linux Projects

You can create projects that generate Linux ELF executable binary files for applications. The project's type is determined by the options you select in the **New Linux/ uClinux Application Project** wizard.

2.2 New Bareboard Project Wizard

The Microcontrollers **New Bareboard Project** wizard presents a series of pages that prompt you for the features and settings to be used when making your program. For example, the devices options lets you select the derivative or board you would like to use. This wizard also helps you specify other settings, such as whether the program executes on an emulator or simulator rather than actual hardware, and the characteristics of the connection that communicates with a hardware target.

This topic describes the various pages that the wizard displays as it assists you in creating a bareboard project. The pages of the wizard can differ based on the project type or execution target.

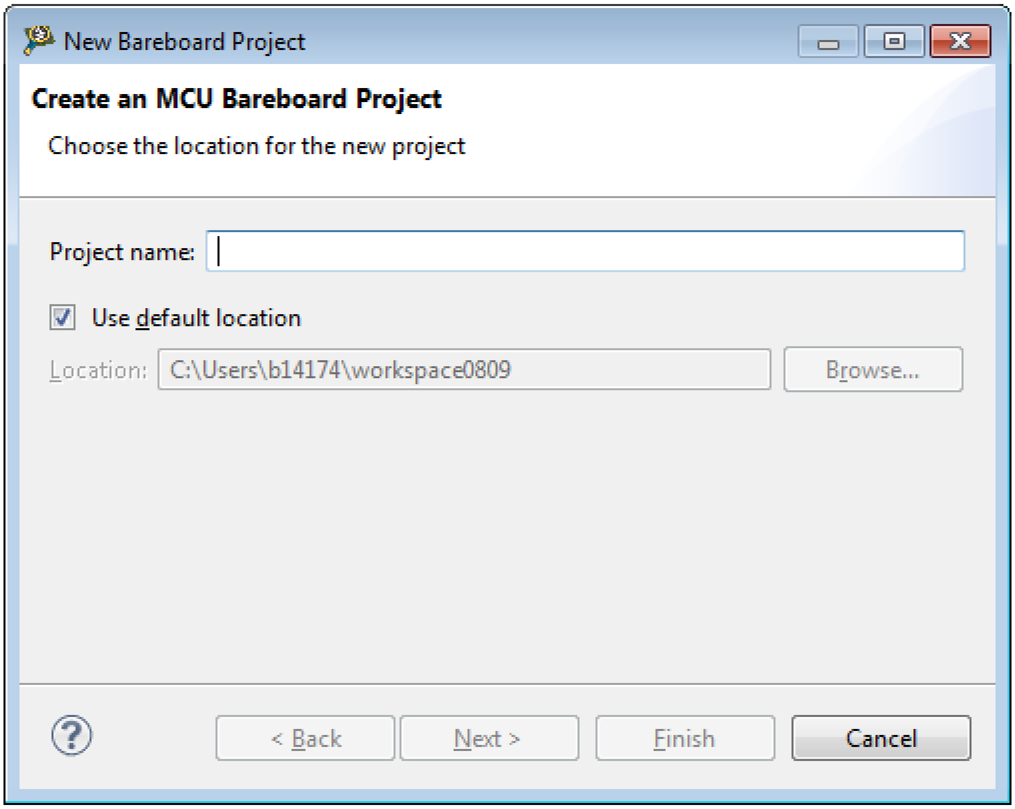
The pages of the **New Bareboard Project** wizard are:

- Create an MCU Bareboard Project Page
- Devices Page
- Connections Page
    - 56800/E DSC derivatives
    - S08/ RS08 derivatives
    - ColdFire derivatives
    - Kinetis derivatives

    - Qorivva derivatives
    - S12Z derivatives
- LSM/DPM Configuration Page
- Power Architecture Core Configuration Page
- Languages Page
    - S08/RS08 derivatives
- Languages and Build Tools Options Page
    - 56800/E (DSC) derivatives
    - Kinetis derivatives
    - Qorivva derivatives
    - S12Z derivatives
- ColdFire Build Options Page
    - ColdFire V1/ColdFire+ V1/Sensors Derivatives
    - ColdFire V2-4e/Vx Derivatives
- Rapid Application Development Page
- C/C++ Options Page

2.2.1 Create an MCU Bareboard Project Page

Use this page to specify the project name and the directory where the project files are located.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name:

☑ Use default location

Location: C:\Users\b14174\workspace0809

Figure 2-1. Create an MCU Bareboard Project Page

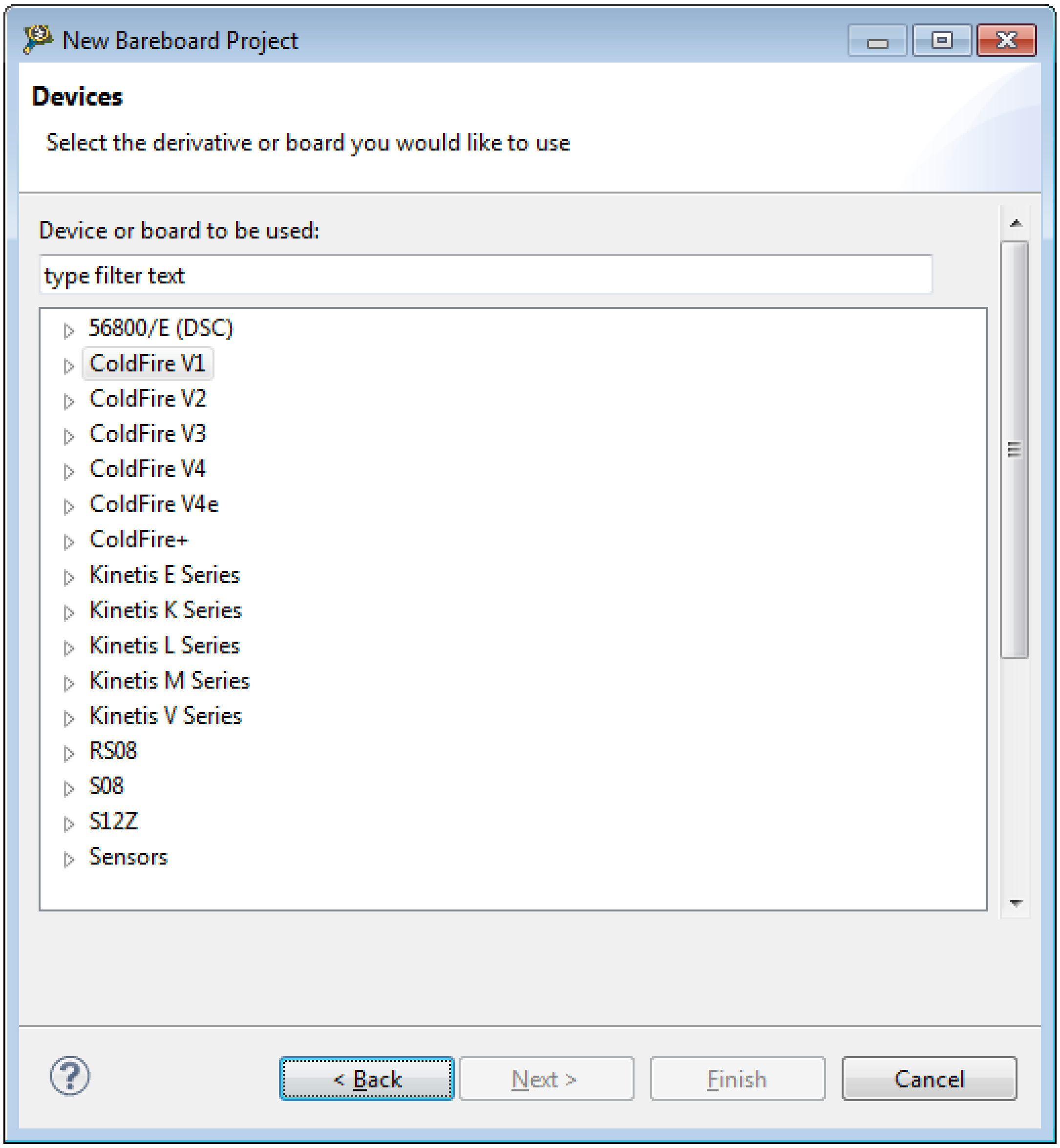
The table below describes the purpose of the various options.

Table 2-1. Create an MCU Bareboard Project Page Settings

Option	Description
Project Name	Enter the name for the new project in this text box. **Note:** Do not use the reserved/special characters/symbols such as < (less than), > (greater than), : (colon), " (double quote), / (forward slash), \ (backslash), | (vertical bar or pipe), ? (question mark), @ (at), * (asterisk) in the project name. sing special characters/symbols in the project name may result in an unexpected behavior.
Use default location	Stores the files required to build the program in the Workbench's current workspace directory. The project files are stored in the default location. Clear the Use default location checkbox and click Browse to select a new location.
Location	Specifies the directory that contains the project files. Click Browse to navigate to the desired directory. This option is available only when Use default location checkbox is clear.

2.2.2 Devices Page

Use this page to select the derivative or board you would like to use.



New Bareboard Project

Devices

Select the derivative or board you would like to use

Device or board to be used:

type filter text

- 56800/E (DSC)
- ColdFire V1
- ColdFire V2
- ColdFire V3
- ColdFire V4
- ColdFire V4e
- ColdFire+
- Kinetis E Series
- Kinetis K Series
- Kinetis L Series
- Kinetis M Series
- Kinetis V Series
- RS08
- S08
- S12Z
- Sensors

Figure 2-2. Devices Page

NOTE

The pages of the wizard change depending on the selected derivative or board.

2.2.3 Connections Page

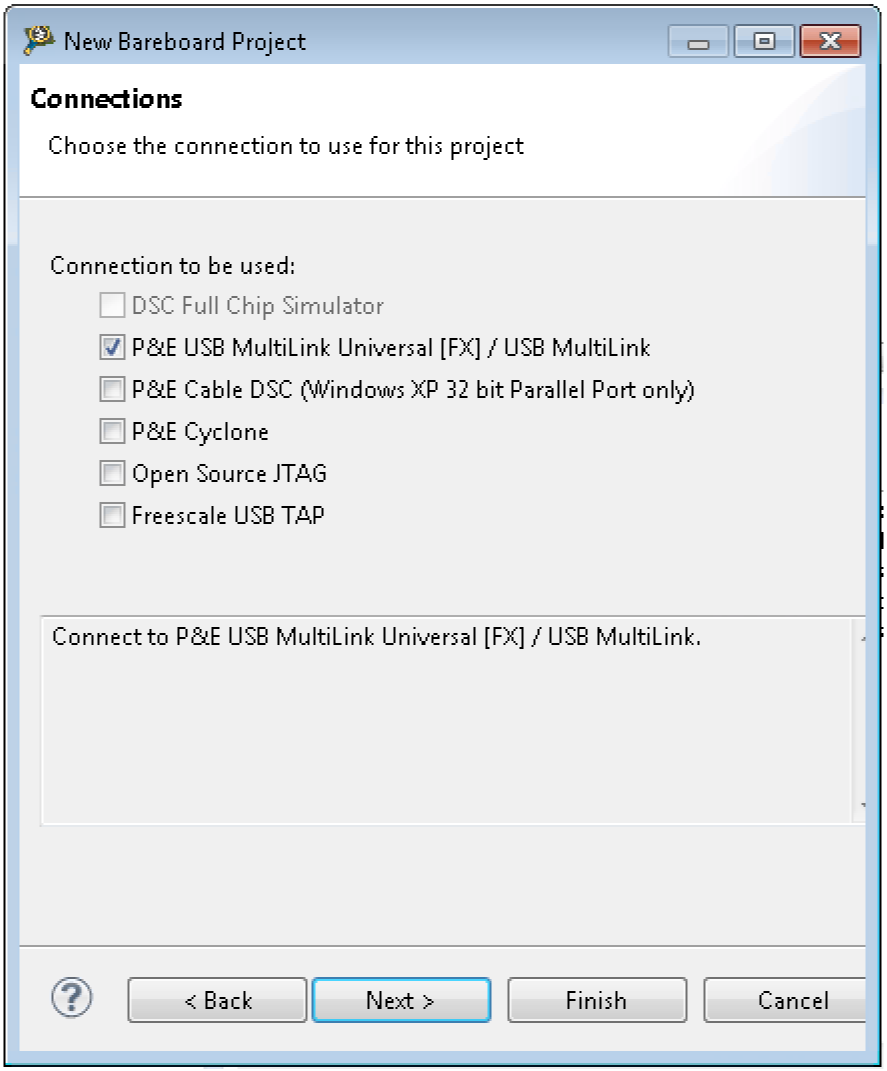
Use this page to select a connection to use for the project. This section explains the connections available for the S08, RS08, ColdFire, Kinetis, DSC, Qorivva and S12Z derivatives.

In this topic the connections page for the following derivatives are listed:

- 56800/E DSC Derivatives
- S08/ RS08 Derivatives
- ColdFire Derivatives
- Kinetis Derivatives
- Qorivva Derivatives
- S12Z Derivatives

2.2.3.1 56800/E DSC derivatives

If you select an *56800/E* derivative or board in the **Devices** page, the **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- □ DSC Full Chip Simulator
- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cable DSC (Windows XP 32 bit Parallel Port only)
- □ P&E Cyclone
- □ Open Source JTAG
- □ Freescale USB TAP

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-3. Connections Page - 56800/ E DSC Derivatives

NOTE

The options appear enabled or grayed out, depending on the derivative or board you selected. For example, the DSC Simulator connection will appear grayed out if you select any of the MC56F844xx, MC56F827xx, MC56F845xx, or MC56F847xx devices.

The table below lists and describes the connections available on the **Connections** page for DSC derivatives.

Table 2-2. Connections Page Settings for 56800/E DSC Derivatives

Option	Description
DSC Full Chip Simulation	Connects to the DSC Full Chip Simulation for simulation of all on-chip peripherals.

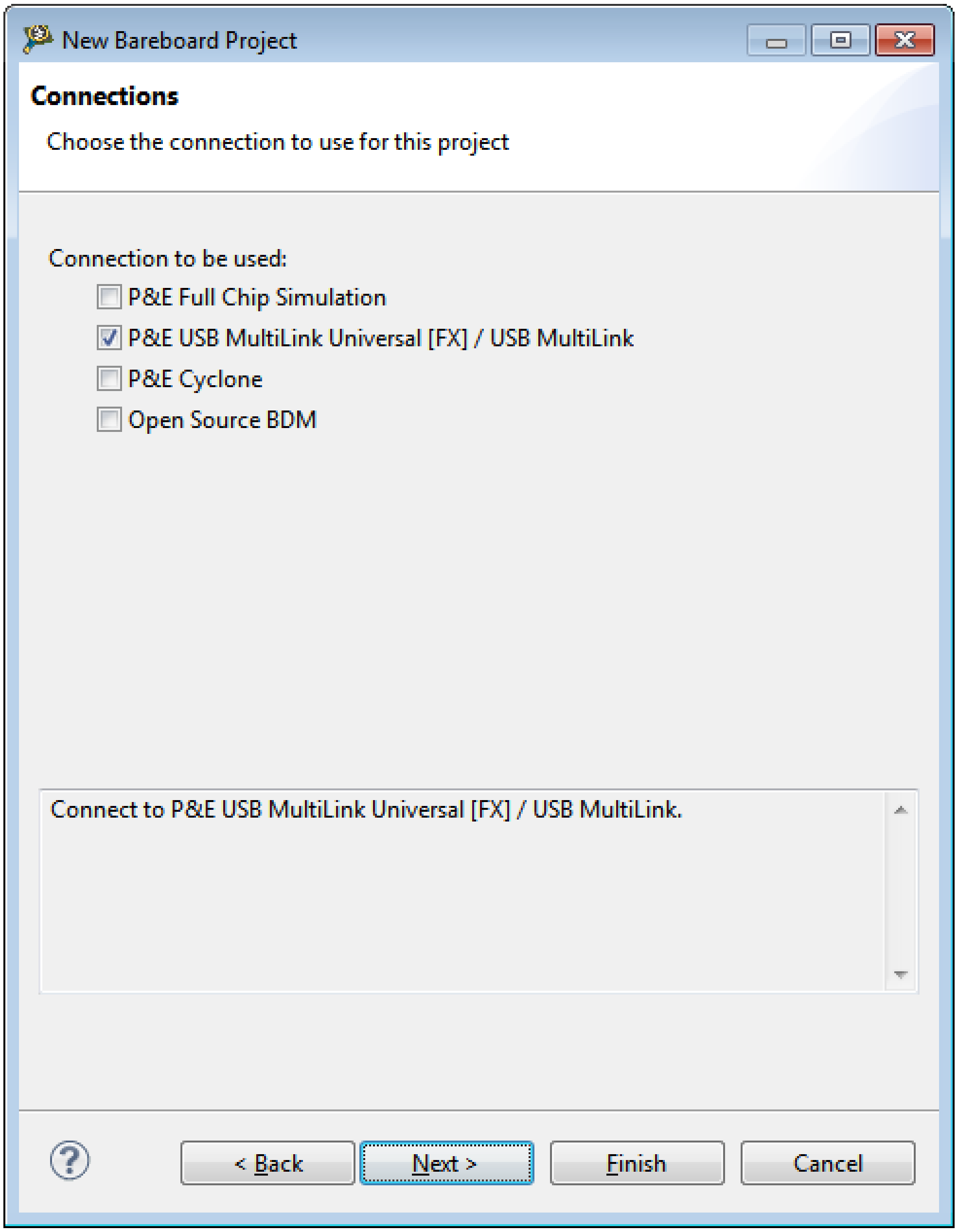
Table continues on the next page...

Table 2-2. Connections Page Settings for 56800/E DSC Derivatives (continued)

Option	Description
P&E USB MultiLink Universal [FX] / USB MultiLink	Connects to the P&E USB MultiLink Universal [FX] / USB MultiLink.
P&E Cable DSC (Windows XP 32 bit Parallel Port only)	Connects to the P&E Cable DSC. This option is supported on Windows XP 32-bit only.
P&E Cyclone	Connects to the Cyclone
Open Source JTAG	Connects to the Open Source JTAG
Freescale USB TAP	Connects to the Freescale USB TAP.

2.2.3.2 S08/ RS08 derivatives

If you select an *S08/RS08* derivative or board in the **Devices** page, the **Connections** page appears.



New Bareboard Project

Connections

Choose the connection to use for this project

Connection to be used:

- ☐ P&E Full Chip Simulation
- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- ☐ P&E Cyclone
- ☐ Open Source BDM

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-4. Connections Page - S08/RS08 Derivatives

NOTE

The options appear enabled or grayed out, depending on the derivative or board you selected.

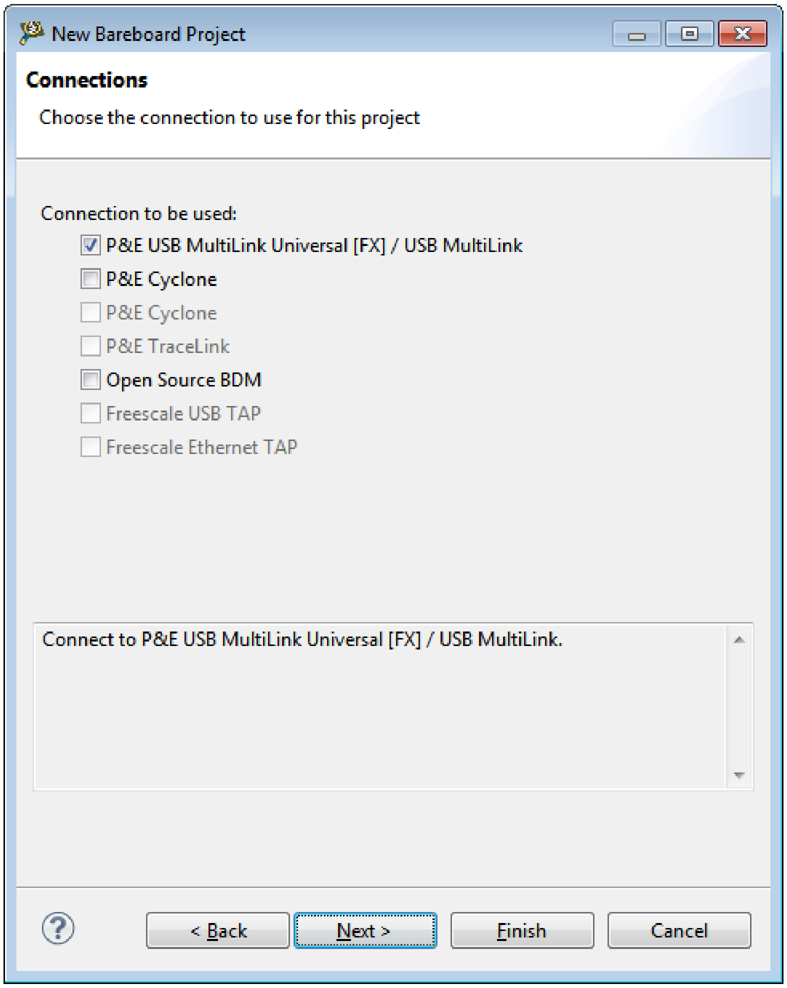
The table below lists and describes the connections available on the **Connections** page for S08/RS08 derivatives.

Table 2-3. Connections Page Settings for HCS08/RS08 Derivatives

Option	Description
P&E Full Chip Simulation	Connects to the P&E Full Chip Simulation for simulation of all on-chip peripherals.
P&E USB MultiLink Universal [FX] / USB MultiLink	Connects to the P&E USB MultiLink Universal [FX] / USB MultiLink.
P&E Cyclone	Connects to the P&E Cyclone (USB).
Open Source BDM	Connects to the Open Source BDM.

2.2.3.3 ColdFire derivatives

If you select a *ColdFire* derivative or board in the **Devices** page, the **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ P&E Cyclone
- □ P&E TraceLink
- □ Open Source BDM
- □ Freescale USB TAP
- □ Freescale Ethernet TAP

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-5. Connections Page - ColdFire Derivatives

NOTE

> The options appear enabled or grayed out, depending on the derivative or board you selected.

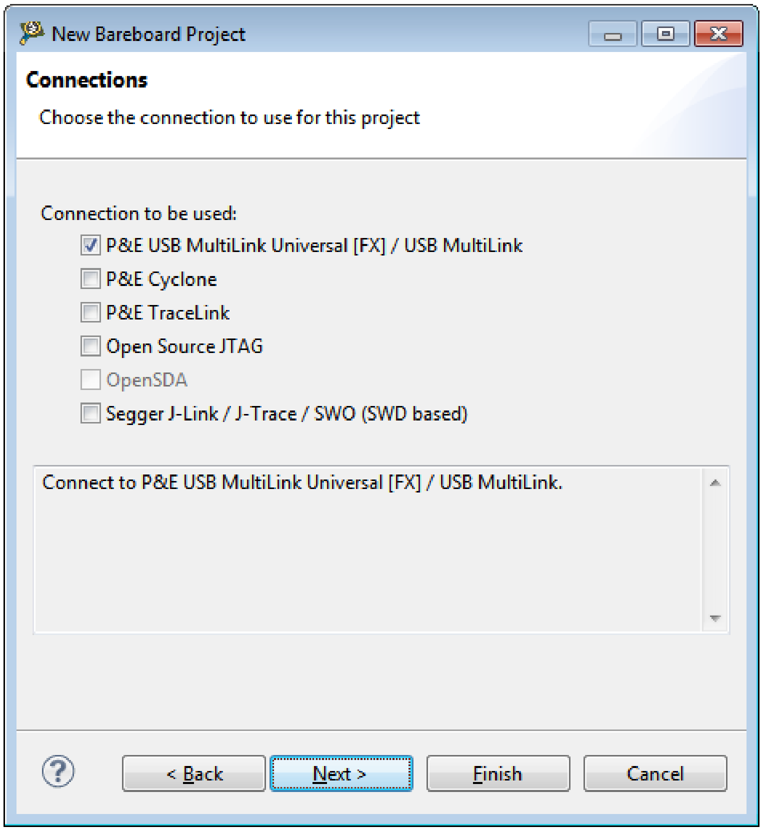
The table below explains the connections available on the **Connections** page for ColdFire derivatives.

Table 2-4. Connections Page Settings for ColdFire Derivatives

Option	Description
P&E USB MultiLink Universal [FX] / USB MultiLink	Connects to the P&E USB MultiLink Universal [FX] / USB MultiLink.
P&E Cyclone	Connects to the P&E Cyclone (USB).
P&E Cyclone	Connects to the P&E Cyclone through the host USB port.
P&E TraceLink	Connects to the P&E TraceLink through the host USB port.
Open Source BDM	Connects to the Open Source BDM.
Freescale USB TAP	Connects to the Freescale USB TAP.
Freescale Ethernet TAP	Connects to the Freescale Ethernet TAP.

2.2.3.4 Kinetis derivatives

If you select a *Kinetis* board or derivative in the **Devices** page, the **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ P&E TraceLink
- □ Open Source JTAG
- □ OpenSDA
- □ Segger J-Link / J-Trace / SWO (SWD based)

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-6. Connections Page - Kinetis Derivatives

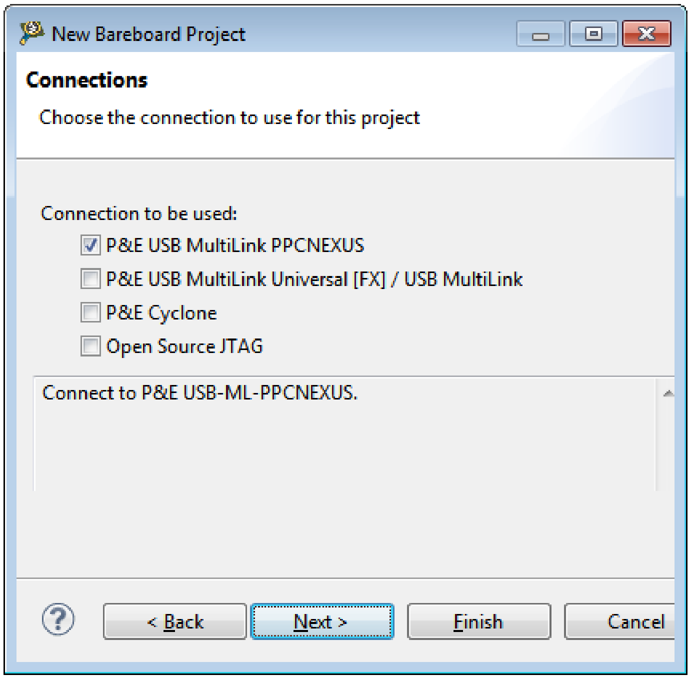
The table below explains the connections available on the **Connections** page.

Table 2-5. Connections Page Settings for Kinetis Derivatives

Option	Description
P&E USB MultiLink Universal [FX] / USB MultiLink	Connects to the P&E USB MultiLink Universal [FX] / USB MultiLink.
P&E Cyclone	Connects to the P&E Cyclone (USB default).
P&E TraceLink	Connects to the P&E TraceLink (USB default).
Open Source JTAG	Connects to the Open Source JTAG.
OpenSDA	Connects to the OpenSDA.
Segger J-Link / J-Trace / SWO (SWD based)	Connects to the Segger J-Link / J-Trace / SWO (SWD based)..

2.2.3.5 Qorivva derivatives

If you select a *Qorivva* board or derivative in the **Devices** page, the **Connections** page appears.



New Bareboard Project

Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink PPCNEXUS
- □ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ Open Source JTAG

Connect to P&E USB-ML-PPCNEXUS.

Figure 2-7. Connections Page - Qorivva Derivatives

The table below explains the connections available on the **Connections** page.

Table 2-6. Connections Page Settings for Qorivva Derivatives

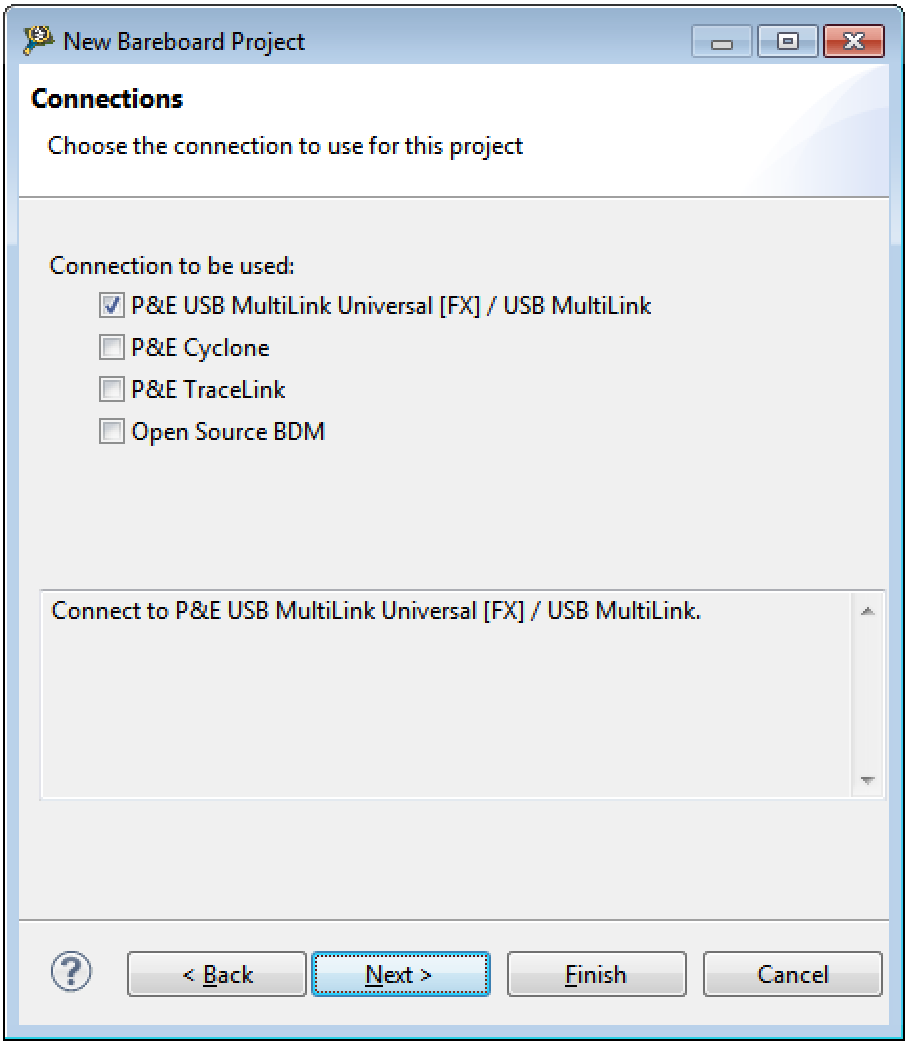
Option	Description
P&E USB MultiLink PPCNEXUS	Connects to the P&E USB MultiLink PPCNEXUS.
P&E USB MultiLink Universal [FX] / USB MultiLink	Connects to the P&E USB MultiLink Universal [FX] / USB MultiLink.
P&E Cyclone	Connects to the P&E Cyclone (USB default).
Open Source JTAG	Connects to the Open Source JTAG.

NOTE

The CodeWarrior debugger supports NEXUS ISTO IEEE 5001-2003 and NEXUS ISTO IEEE 5001-2010 for Qorivva families.

2.2.3.6 S12Z derivatives

If you select an *S12Z* board or derivative in the **Devices** page, the **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ P&E TraceLink
- □ Open Source BDM

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-8. Connections Page - S12Z Derivative

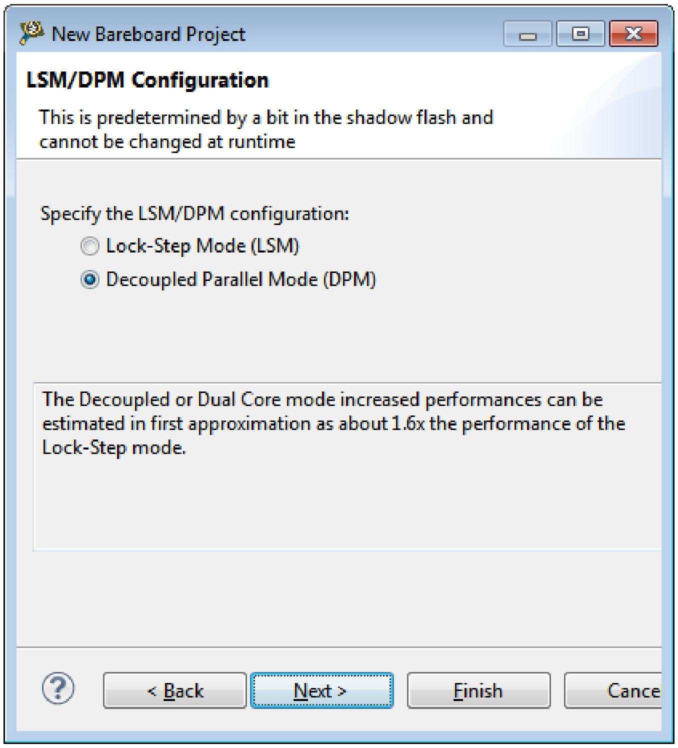
The table below explains the connections available on the **Connections** page.

Table 2-7. Connections Page Settings for S12Z Derivative

Option	Description
P&E USB MultiLink Universal [FX] / USB MultiLink	Connects to the P&E USB MultiLink Universal [FX] / USB MultiLink.
P&E Cyclone	Connects to the P&E Cyclone(USB default).
P&E TraceLink	Connect to P&E TraceLink (USB default).
Open Source BDM	Connects to the Open Source BDM.

2.2.4 LSM/DPM Configuration Page

Use this page to select the **Lock-Step Mode (LSM)** and **Decoupled Parallel Mode (DPM)** configuration. This is predetermined by a bit in the shadow flash and cannot be changed at runtime.



LSM/DPM Configuration

This is predetermined by a bit in the shadow flash and cannot be changed at runtime

Specify the LSM/DPM configuration:

○ Lock-Step Mode (LSM)

● Decoupled Parallel Mode (DPM)

The Decoupled or Dual Core mode increased performances can be estimated in first approximation as about 1.6x the performance of the Lock-Step mode.

Figure 2-9. DPM/LSM Configuration Page

NOTE

This page appears only for *Qorivva's* MPC56xxK and MPC56xxL families.

The table below explains the connections available on the **DPM/LSM Configuration** page.

Table 2-8. DPM/LSM Configuration Page Settings

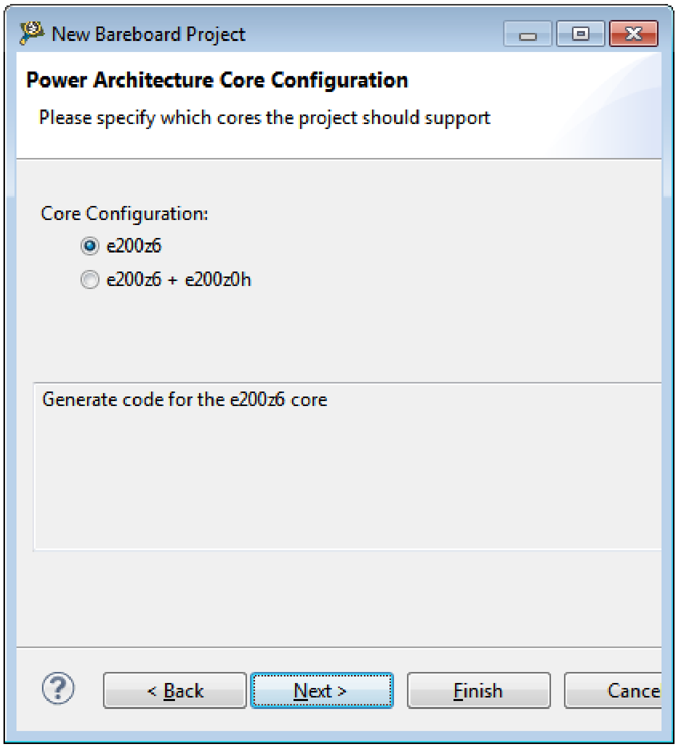
Option	Description
Lock-Step Mode (LSM)	Intended for safety critical systems that require redundancy.
Decoupled Parallel Mode (DPM)	Intended to increase performances that can be estimated in first approximation as about 1.6x the performance of the LS mode.

NOTE

Many devices in the 55xx/56xx family are multicore devices (multiple e200 and eTPU cores). The debugger shall provide multicore debugging for the 55xx/56xx devices having multicore built in. This extends to both lock-step mode (LS mode or LSM) as well to Decoupled Parallel Mode (DP mode or DPM).

2.2.5 Power Architecture Core Configuration Page

Use this page to provide Power Architecture configuration when creating your project.



New Bareboard Project

Power Architecture Core Configuration

Please specify which cores the project should support

Core Configuration:

● e200z6

○ e200z6 + e200z0h

Generate code for the e200z6 core

Figure 2-10. Power Architecture Core Configuration Page

NOTE

This page appears only for Qorivva's MPC5668E/G Family derivatives.

The following table describes the options available for the Power Architecture Core Configuration page.

Table 2-9. Add Files Page Settings

Option	Description
e200z6	Select to generate core for e200z6 core.
e200z6 + e200z0h	Select to generate core for e200z6 and e200z0h core.

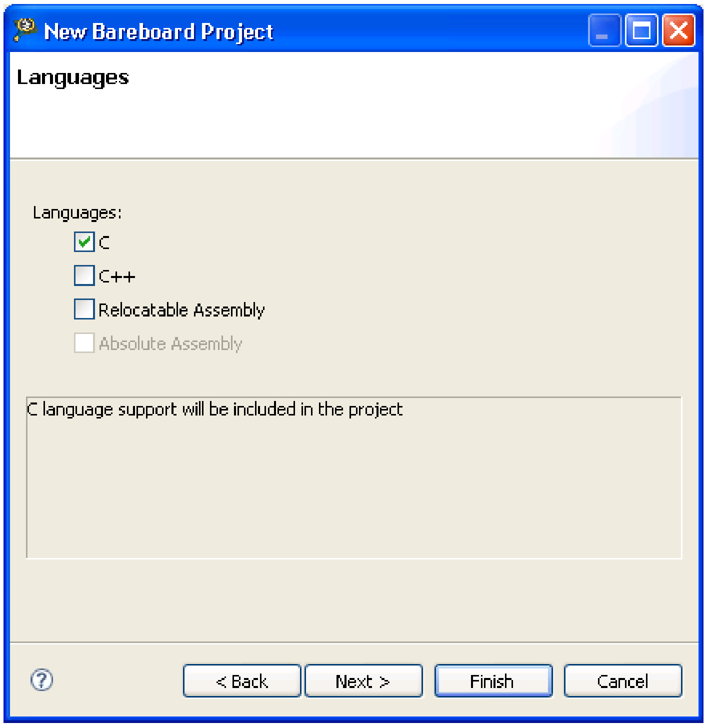
2.2.6 Languages Page

Use this page to select the programming language that you want to use for writing the program's source code. In this topic:

- S08/RS08 derivatives

2.2.6.1 S08/RS08 derivatives

If you select an *S08/RS08* derivative, the languages page appears. You can make multiple selections, creating the code in multiple formats.



Languages:
☑ C
☐ C++
☐ Relocatable Assembly
☐ Absolute Assembly

C language support will be included in the project

Figure 2-11. Languages page - S08/RS08 derivative

NOTE

The options appear enabled or grayed out, depending on the derivative or board you selected.

The table below explains the options available on this page.

Table 2-10. Languages page settings - S08/RS08 derivatives

Group / Option	Description
C	Checking the **C** checkbox sets up your application with ANSI C-compliant startup code, and initializes global variables.
C++	Checking the **C++** checkbox sets up your application with ANSI C++ startup code, and performs global class object initialization.
Relocatable Assembly	Checking the **Relocatable** checkbox enables you to split up the application into multiple assembly source files. The source files are linked together using the linker.
Absolute Assembly	Checking the **Absolute Assembly** checkbox enables you to use only one single assembly source file with absolute assembly. There is no support for relocatable assembly or linker.

NOTE

The option you select also sets up default compiler/linker options for the toolchain. For example, if you plan to use the C language in your source code files, check the **C** checkbox. If you plan to write the program using C++, check the **C++** checkbox.

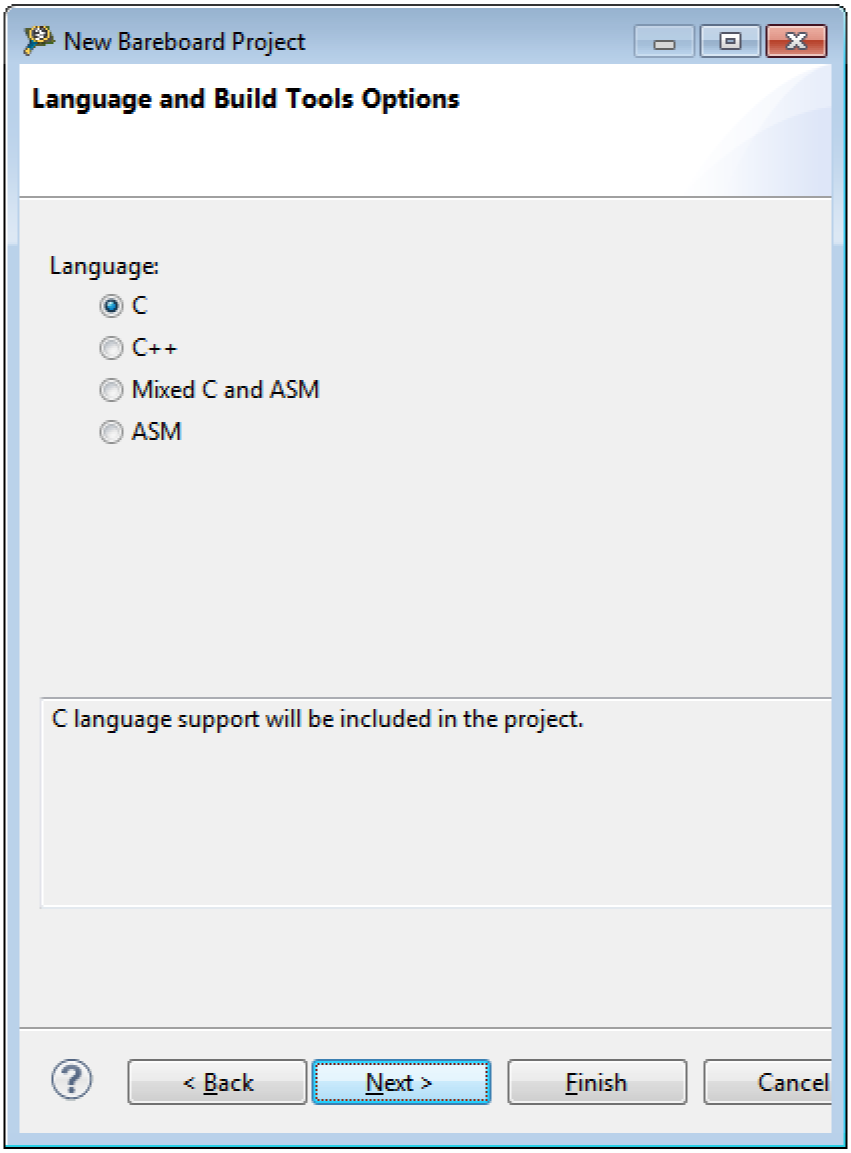
2.2.7 Languages and Build Tools Options Page

Use this page to select the programming language, build tools options, and float point format support that you want to use when writing the program's source code.

- 56800/E (DSC) derivatives
- Kinetis Derivatives
- Qorivva derivatives
- Languages Page

2.2.7.1 56800/E (DSC) derivatives

Use this page to select the programming language that you want to use when writing the program's source code for 56800/E (DSC) or S12Z derivative. You can make only single selection, creating the code in single formats.



Language and Build Tools Options

Language:

● C

○ C++

○ Mixed C and ASM

○ ASM

C language support will be included in the project.

Figure 2-12. Languages page - DSC/S12Z derivatives

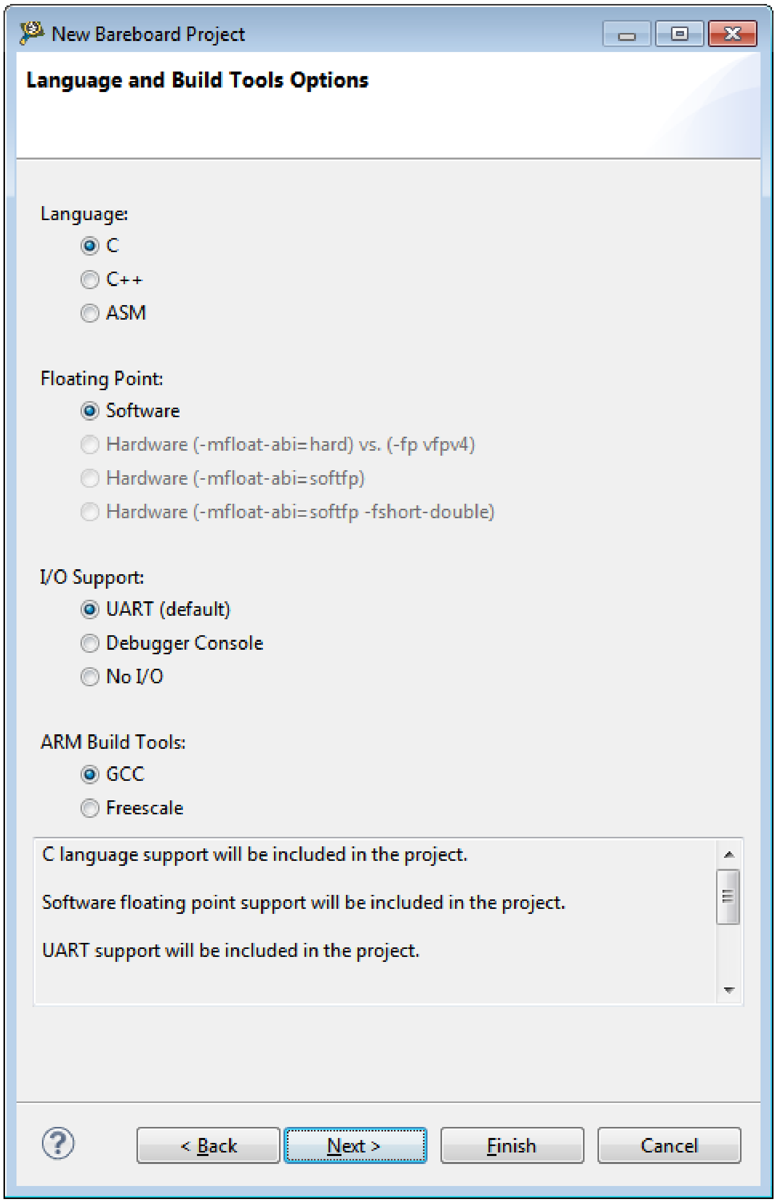
The table below explains the options available on this page.

Table 2-11. Languages page settings - DSC/S12Z derivatives

Option	Description
C	Select this option to include C language support in your project.
C++	Select this option to include C++ language support in your project.
Mixed C and ASM	Select this option to include mixed C/assembly language support in your project.
Simple Assembly	Select this option to include assembly language support in your project.

2.2.7.2 Kinetis derivatives

Use this page to select the programming language that you want to use when writing the program's source code for Kinetis derivative. You can make only single selection, creating the code in single formats.



Language and Build Tools Options

Language:

- ● C
- ○ C++
- ○ ASM

Floating Point:

- ● Software
- ○ Hardware (-mfloat-abi=hard) vs. (-fp vfpv4)
- ○ Hardware (-mfloat-abi=softfp)
- ○ Hardware (-mfloat-abi=softfp -fshort-double)

I/O Support:

- ● UART (default)
- ○ Debugger Console
- ○ No I/O

ARM Build Tools:

- ● GCC
- ○ Freescale

C language support will be included in the project.

Software floating point support will be included in the project.

UART support will be included in the project.

Figure 2-13. Language and Build Tools Options page - Kinetis derivatives

NOTE

Availability of the options appearing on the page, depends on the board or derivative selected.

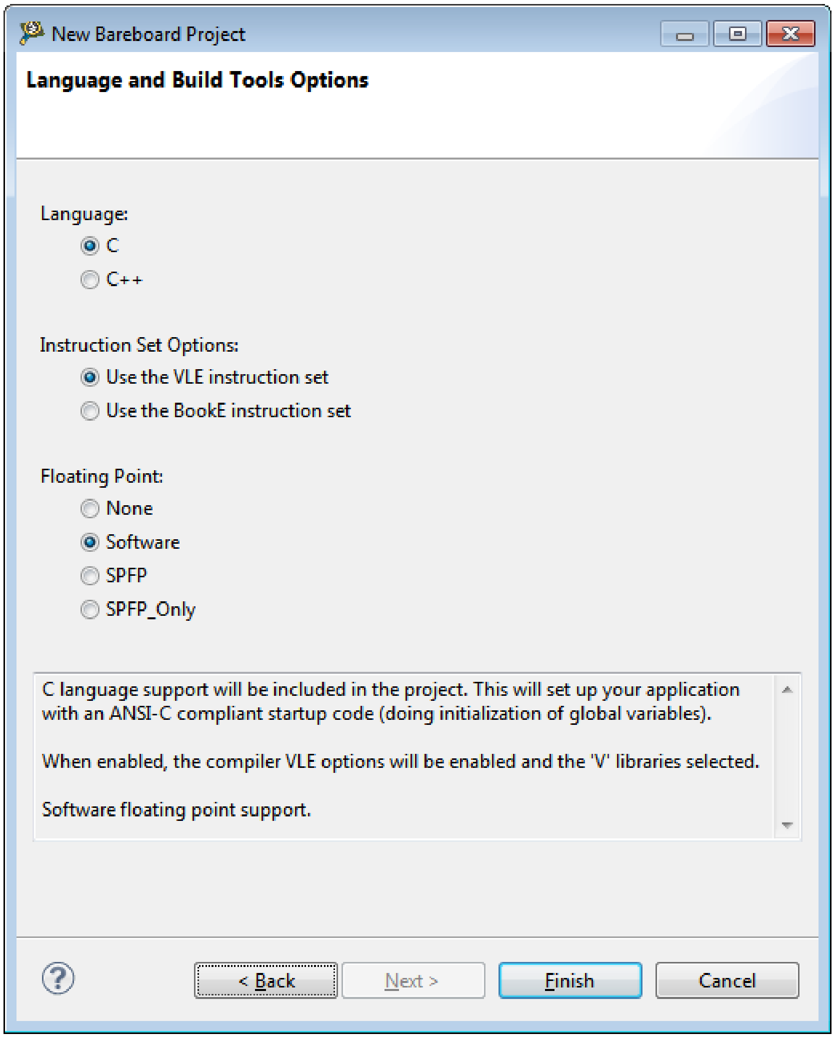
The following table explains the options available on this page.

Table 2-12. Language and Build Tools Options Page settings - Kinetis derivatives

Option	Description
Language	Select this option to set up your application with ANSI C-compliant startup code, and initializes global variables.
C	
C++	Select this option to set up your application with ANSI C++ startup code, and performs global class object initialization.
ASM	Select this option to include assembly language support in your project.
Floating Point	Select this option to include Software floating point support in the project.
Software	
Hardware (`-mfloat-abi=hard) vs. (-fp vfpv4`)	Select to support hardware floating point. Using this option GCC build tools generates the code using hardware floating-point instructions and uses FPU-specific calling convention `-mfloat-abi=hard`. For Freescale Build tools, this option allows performing single precision float operations through SPFPU hardware instructions support: (`-fp vfpv4`).
Hardware (`-mfloat-abi=softfp`)	Select to support hardware floating point. This option allows GCC build tools to generate code using hardware floating-point instructions, but still uses the soft-float calling conventions.
Hardware (`-mfloat-abi=softfp -fshort-double`)	Select to support hardware floating point. Using this option GCC build tools generates code using hardware floating-point instructions, but still uses the soft-float calling conventions. Also, use same size for `double` as for `float`. **WARNING** : The `-fshort-double` switch causes GCC to generate code that is not binary compatible with code generated without that switch. Use it to conform to a non-default application binary interface.
I/O Support	Configures how the library deals with the console (e.g. printf() or puts()). With \`UART' it uses the physical serial device and connection.
UART (default)	
Debugger Console	Configures how the library deals with the console (e.g. printf() or puts()). With \`Debugger Console' the library uses a virtual connection with the debugger (also known as \`semi hosting').
No I/O	No Console Support.
ARM Build Tools	Select this option to use GCC build tools for the project.
GCC	
Freescale	Select this option to use Freescale build tools for the project.

2.2.7.3 Qorivva derivatives

Use this page to select the programming language that you want to use when writing the program's source code for Qorivva derivatives. The following figure shows the **Language and Build Tools Options** Page for the Qorivva derivatives.



Language:
- ● C
- ○ C++

Instruction Set Options:
- ● Use the VLE instruction set
- ○ Use the BookE instruction set

Floating Point:
- ○ None
- ● Software
- ○ SPFP
- ○ SPFP_Only

C language support will be included in the project. This will set up your application with an ANSI-C compliant startup code (doing initialization of global variables).

When enabled, the compiler VLE options will be enabled and the 'V' libraries selected.

Software floating point support.

Figure 2-14. Language and Build Tools Options page - Qorivva derivatives

NOTE

Availability of the options appearing on the page, depends on the board or derivative selected.

The following table explains the options available on this page.

Table 2-13. Language and Build Tools Options page settings - Qorivva derivatives

Option	Description
Language	
C	Checking the **C** checkbox sets up your application with ANSI C-compliant startup code, and initializes global variables.

Table continues on the next page...

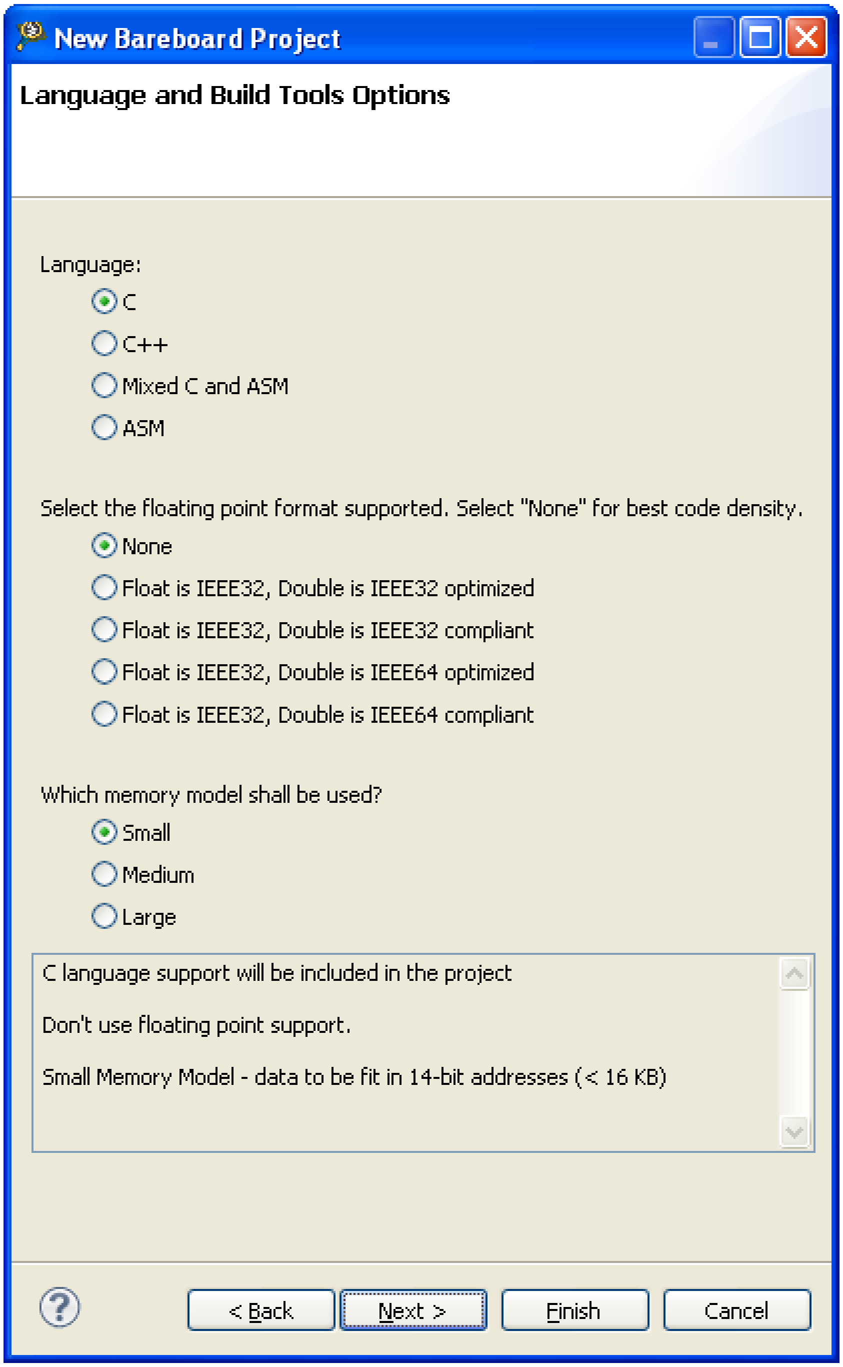
Table 2-13. Language and Build Tools Options page settings - Qorivva derivatives (continued)

Option	Description
C++	Checking the **C++** checkbox sets up your application with ANSI C++ startup code, and performs global class object initialization.
Instruction Set Options	
Use the VLE instruction set	Check to enable the compiler VLE options and select the V libraries. This checkbox is ignored, if the selected processor does not support VLE.
Use the BookE instruction set	Check to enable the compiler BookE options.
Floating Point	
None	Select if you do not want to include floating point support; gives best code density.
Software	Select if you want to include software floating point support.
SPFP	Select if you want single precision floating point support for the project. It performs single precision float operations through SPE-EFPU hardware instruction support and performs double precision float operations by utilizing the software emulation library.
SPFP_Only	Select if you want single precision floating point only support for the project. The compiler considers 'double' and 'long double' data types as single precision 'float' data type.

2.2.7.4 S12Z derivatives

Use this page to select the programming language that you want to use when writing the program's source code for S12Z derivative.

The following figure shows the **Language and Build Tools Options** page for S12Z derivatives.



Language and Build Tools Options

Language:

- ● C
- ○ C++
- ○ Mixed C and ASM
- ○ ASM

Select the floating point format supported. Select "None" for best code density.

- ● None
- ○ Float is IEEE32, Double is IEEE32 optimized
- ○ Float is IEEE32, Double is IEEE32 compliant
- ○ Float is IEEE32, Double is IEEE64 optimized
- ○ Float is IEEE32, Double is IEEE64 compliant

Which memory model shall be used?

- ● Small
- ○ Medium
- ○ Large

C language support will be included in the project

Don't use floating point support.

Small Memory Model - data to be fit in 14-bit addresses (< 16 KB)

Figure 2-15. Language and Build Tools Options page - S12Z derivatives

The following table lists the description of the options available in the **Language and Build Tools Options** page for S12Z derivatives.

Table 2-14. Language and Build Tools Options page settings - S12Z derivatives

Option	Description
Language	

Table continues on the next page...

Table 2-14. Language and Build Tools Options page settings - S12Z derivatives (continued)

Option	Description
C	Check this checkbox to include C language support in the project.
C++	Check this checkbox to include C++ language support in the project.
Mixed C and ASM	Check this checkbox to include Mixed C/Asm language support in the project.
ASM	Check this checkbox to include assembly language support in the project.
Select the floating point format supported. Select "None" for best code density.	Select this option if you do not want to include floating point support; gives best code density.
None	
Float is IEEE32, Double is IEEE32 optimized	Select this option to use all float and double variables as 32-bit/IEEE32. Library is optimized, but lose some of the IEEE754 standard compliance.
Float is IEEE32, Double is IEEE32 compliant	Select this option to use all float and double variables as 32-bit/IEEE32. Library is IEEE754 compliant but loses some of the performance.
Float is IEEE32, Double is IEEE64 optimized	Select this option to use float variables as 32-bit/IEEE32 and double variables as 64-bit/IEEE64. Library is optimized, but loses some of the IEEE754 standard compliance.
Float is IEEE32, Double is IEEE64 compliant	Select this option to use all float variables as 32-bit/IEEE32 and double variables are 64-bit/IEEE64. Library is IEEE754 compliant, but loses some of the performance.
Which memory model shall be used?	Select this option to enable the Small Memory Model. In this model data fits in the 14-bit addresses (< 16 KB).
Small	
Medium	Select this option to enable the Medium Memory Model. In this model data fits in 18-bit addresses (< 256 KB).
Large	Select this option to enable the Large Memory Model. In this model data fits in 24-bit addresses.

NOTE

The IEEE formats recognize several special bit patterns for special values. The number 0 (zero) is encoded by the bit pattern consisting of zero bits only. Other special values such as "Not a number", "infinity", -0 (minus zero) and denormalized numbers do exist. Refer to the IEEE standard documentation for details. Except for the 0 (zero) and -0 (minus zero) special formats, not all special formats may be supported for specific backends.

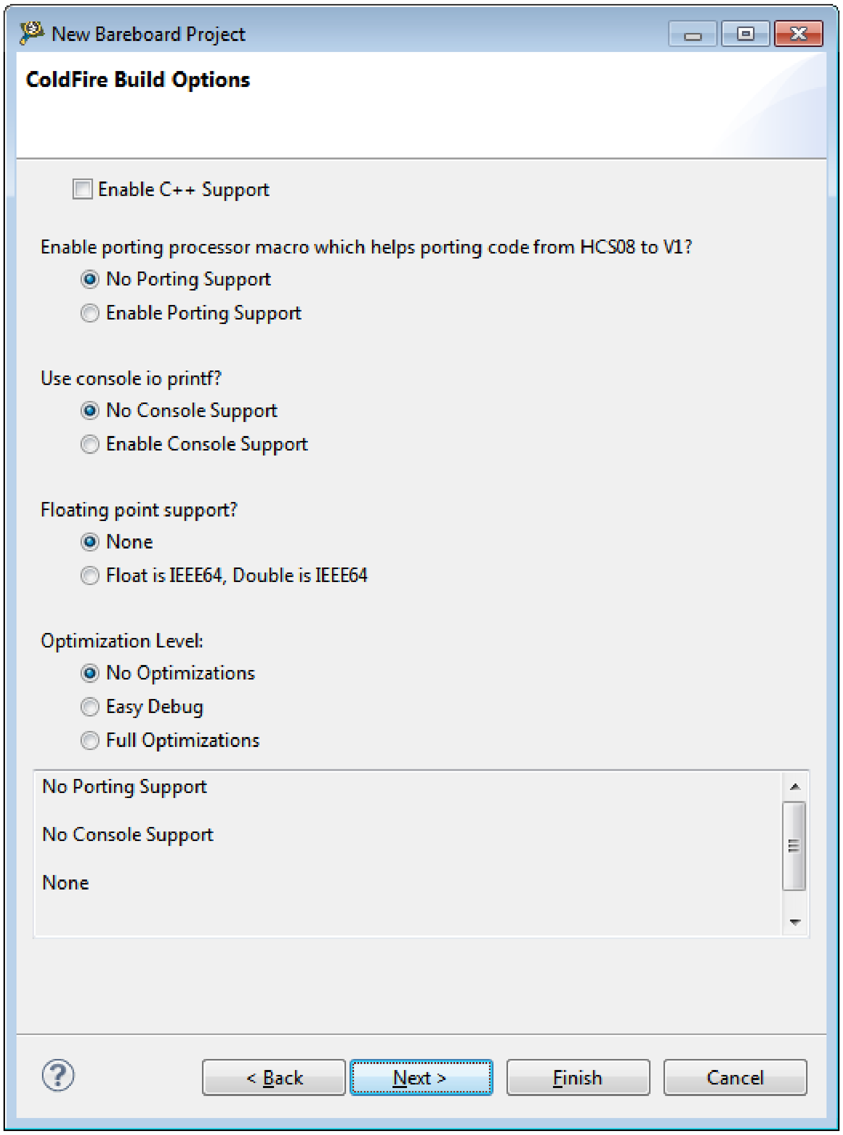
2.2.8 ColdFire Build Options Page

Use this page to select the build tool options for your ColdFire projects. In this topic:

- ColdFire V1/ColdFire+ V1/Sensors Derivatives
- ColdFire V2-4e/Vx Derivatives

2.2.8.1 ColdFire V1/ColdFire+ V1/Sensors Derivatives

You can use this page to enable C++, porting processor macro, console, floating point support, and optimization level for *ColdFire V1/ColdFire+ V1/Sensors* derivatives.



New Bareboard Project

ColdFire Build Options

□ Enable C++ Support

Enable porting processor macro which helps porting code from HCS08 to V1?

● No Porting Support
○ Enable Porting Support

Use console io printf?

● No Console Support
○ Enable Console Support

Floating point support?

● None
○ Float is IEEE64, Double is IEEE64

Optimization Level:

● No Optimizations
○ Easy Debug
○ Full Optimizations

No Porting Support

No Console Support

None

Figure 2-16. ColdFire Build Options Page - ColdFire V1/ColdFire+ V1/ Sensors

The table below explains the options available on this page.

Table 2-15. Build Options Page Settings - ColdFire V1/ColdFire+ V1/ Sensors

Option	Description
Enable C++ Support	Check to enable C++ support
No Porting Support	Select to disable the porting processor macro that helps porting code from HCS08 to V1
Enable Porting Support	Select to enable the porting processor macro that helps porting code from HCS08 to V1
No Console Support	Select to disable the console support
Enable Console Support	Select to enable the console support
None	Select to disable the floating point support
Float is IEEE64, double is IEEE64	Select to enable the floating point support

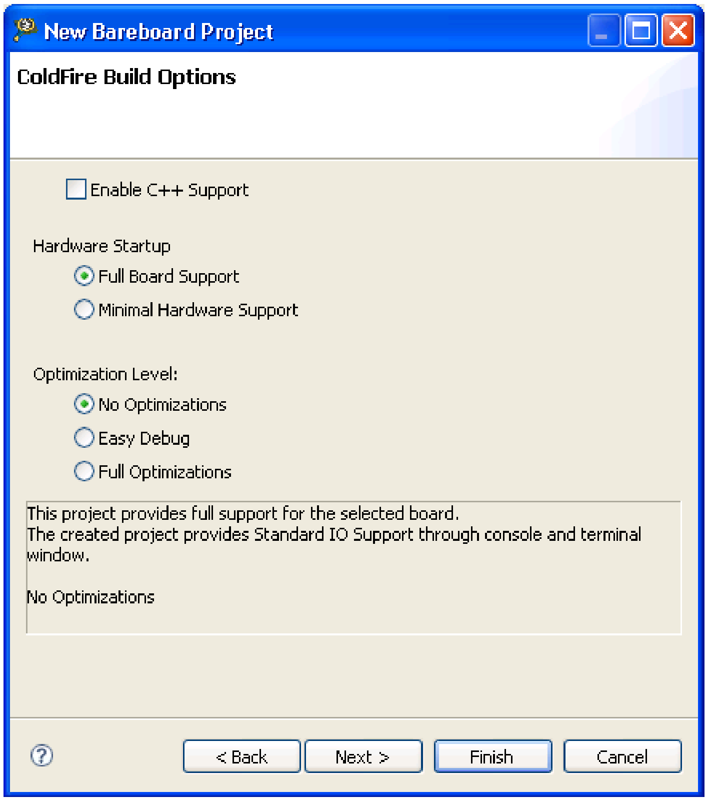
Table continues on the next page...

Table 2-15. Build Options Page Settings - ColdFire V1/ColdFire+ V1/ Sensors (continued)

Option	Description
No Optimizations	Select to disable the optimization level
Easy Debug	Select to enable the Level 1 code size optimizations plus register coloring and peephole
Full Optimizations	Select to enable the full optimizations

2.2.8.2 ColdFire V2-4e/Vx Derivatives

Use this page to enable C++, porting processor macro, hardware startup, and optimization level for *ColdFire V2/V3/V4/V4e/Vx Evaluation Boards/Vx Tower Boards* derivatives.



ColdFire Build Options

□ Enable C++ Support

Hardware Startup

● Full Board Support

○ Minimal Hardware Support

Optimization Level:

● No Optimizations

○ Easy Debug

○ Full Optimizations

This project provides full support for the selected board.
The created project provides Standard IO Support through console and terminal window.

No Optimizations

Figure 2-17. ColdFire Build Options Page - ColdFire V2-4e/Vx Derivatives

The table below explains the options available on this page.

Table 2-16. Build Options Page Settings - ColdFire V2-4e/Vx

Option	Description
Enable C++ Support	Check to enable C++ support

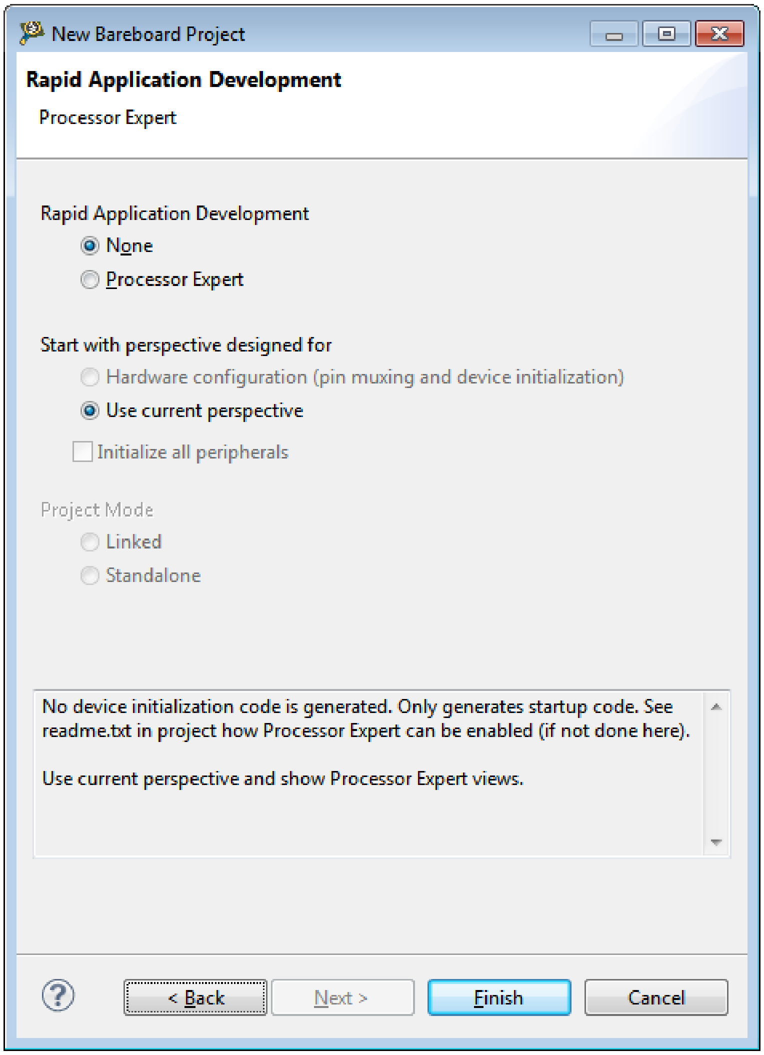
Table continues on the next page...

Table 2-16. Build Options Page Settings - ColdFire V2-4e/Vx (continued)

Option	Description
Full Board Support	Select to provide full support for the selected board. The created project provides standard input output support through console and terminal window.
Minimal Hardware Support	Select if you do not want to provide board initialization support. The project can be customized or used with the Instruction Set Simulator. The standard input output support is enabled for the Console build target. However, you need to enable UART support for standard input output support through UART, by providing the correct system clock.
No Optimizations	Select to disable optimization level
Easy Debug	Select to enable Level 1 code size optimizations plus register coloring and peephole
Full Optimizations	Select to enable full optimizations

2.2.9 Rapid Application Development Page

Use this page to provide Rapid Application Development (RAD) support when writing your program.



Rapid Application Development

Processor Expert

Rapid Application Development

● None

○ Processor Expert

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

Project Mode

○ Linked

○ Standalone

No device initialization code is generated. Only generates startup code. See readme.txt in project how Processor Expert can be enabled (if not done here).

Use current perspective and show Processor Expert views.

Figure 2-18. Rapid Application Development Page

Select one of the available RAD options to set up special views in the IDE where you can rapidly configure peripheral devices on the MCU, or pick from a library of field-tested code modules that can implement various device services such as timer interrupts, or a high speed serial interface.

NOTE

This page is not available for Qorivva, and some ColdFire derivatives.

NOTE

For more information on how to use the features of Processor Expert, refer to the Processor Expert User Guide.

The table below shows the various RAD options available and their purpose.

Table 2-17. Rapid Application Development Page Settings

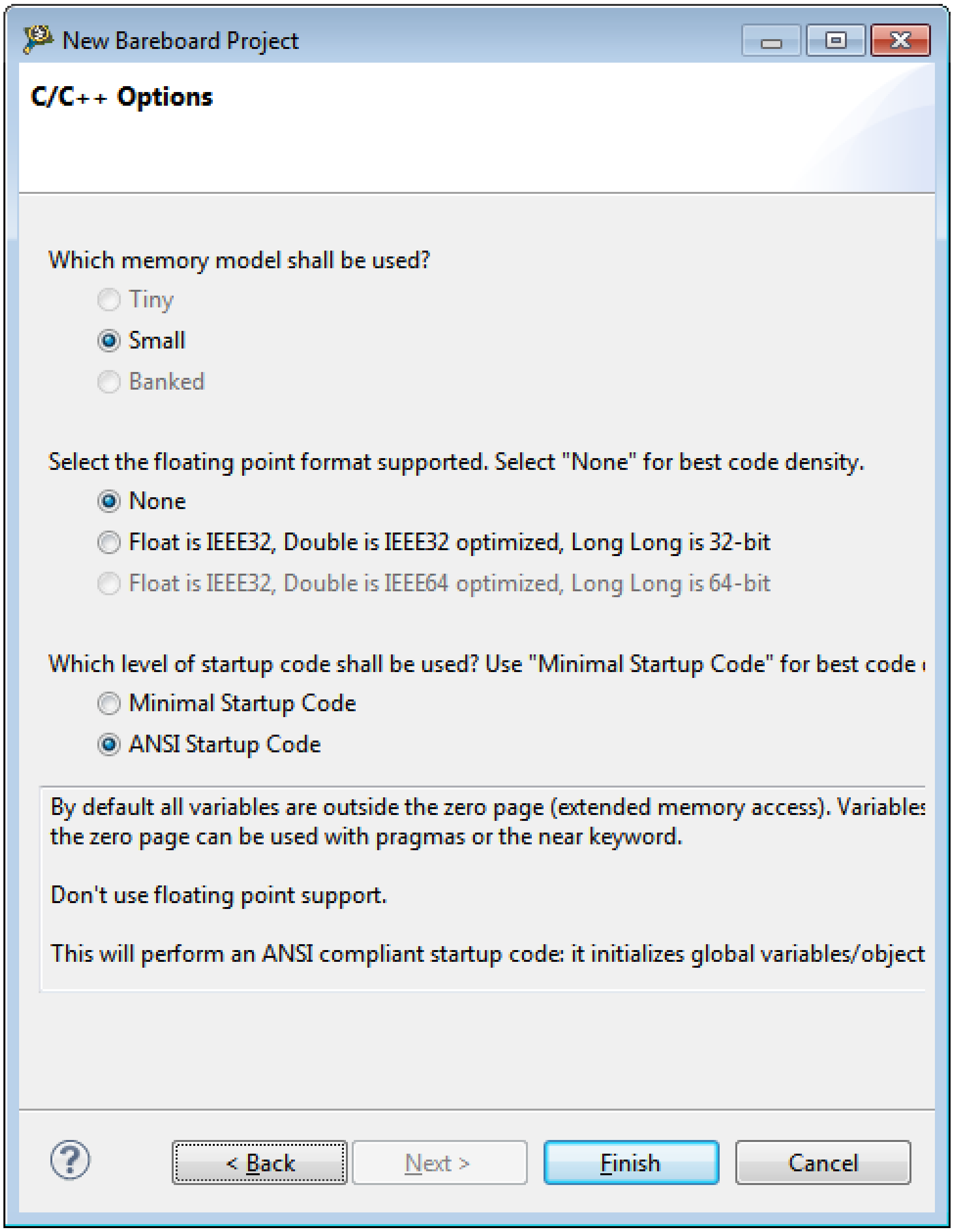
Option	Description
Rapid Application Development	
None	No RAD support provided. The wizard's default startxx.c file sets up the MCU's stack, its memory management unit (if any) and the C/C++ language's runtime.
Processor Expert	The wizard provides views in the C/C++ Perspective that lets you set up the MCU's interrupts, vector table and device initialization. It also provides you with a choice of configurable support modules that implement software services on various MCU peripherals.
Start with perspective designed for	
Hardware configuration (pin muxing and device initialization)	Select this option to enable hardware perspective for pin muxing and peripheral configuration.
Use current perspective	Select this option to use current perspective and show the Processor Expert views.
Initialize all peripherals	Select this option to initialize all peripherals.

2.2.10 C/C++ Options Page

Use this page to select the level of startup code you want to produce, the memory model, and the appropriate floating point format support.

NOTE

This page is available only for S08/RS08 derivatives.



New Bareboard Project

C/C++ Options

Which memory model shall be used?

○ Tiny

● Small

○ Banked

Select the floating point format supported. Select "None" for best code density.

● None

○ Float is IEEE32, Double is IEEE32 optimized, Long Long is 32-bit

○ Float is IEEE32, Double is IEEE64 optimized, Long Long is 64-bit

Which level of startup code shall be used? Use "Minimal Startup Code" for best code

○ Minimal Startup Code

● ANSI Startup Code

By default all variables are outside the zero page (extended memory access). Variables
the zero page can be used with pragmas or the near keyword.

Don't use floating point support.

This will perform an ANSI compliant startup code: it initializes global variables/object

Figure 2-19. C/C++ Options Page

NOTE

The availability of the options appearing on the page, depends on the derivative or board selected.

The table below explains the options available on this page.

Table 2-18. C/C++ Options Page Settings

Option	Description
Tiny	Assumes that data pointers have 8-bit addresses unless explicitly specified with the keyword `__far`.

Table continues on the next page...

Table 2-18. C/C++ Options Page Settings (continued)

Option	Description
Small	Use the Small memory model if both the code and the data fit into the 64-kilobyte address space. By default, all variables and functions are accessed with 16-bit addresses. The compiler supports banked functions or paged variables in this memory model, but all accesses must be explicitly handled.
Banked	Banked memory model uses banked function calls by default, but the default data access is still 16-bit. Because the overhead of the far function call is not very large, this memory model suits all applications with more than 64-kilobytes of code. Data paging can be used, however all far objects and pointers to them must be specially declared.
None	Select for the best code intensity.
Float is IEEE32, Double is IEEE32 optimized	All float and double variables are 32-bit IEEE32. Library is optimized, but loses some of the IEEE754 standard compliance.
Float is IEEE32, Double is IEEE64 optimized	All float variables are 32-bit/IEEE32 and Double variables are 64-bit/IEEE64. Library is optimized, but loses some of the IEEE754 standard compliance.
Minimal startup code	Produces the best code density. The startup code initializes the stack pointer and calls the main function. No initialization of global variables is done, giving you the best speed/code density and a fast startup time. The application code must address variable initialization. ANSI requires variable initialization and therefore this option is not ANSI compliant.
ANSI startup code	Initializes global variables/objects and calls the application main routine.

2.3 New Linux/uClinux Application Project Wizard

When you start the Microcontrollers **New Linux/uClinux Application Project** wizard, it presents you with a sequence of pages that prompt you for the features and settings to be used when making your program. For example, the devices options lets you select the ColdFire derivative or board you would like to use. Other options let you to specify other settings, such as whether the program executes on an emulator or simulator rather than actual hardware, and the characteristics of the connection that communicates with a hardware target.

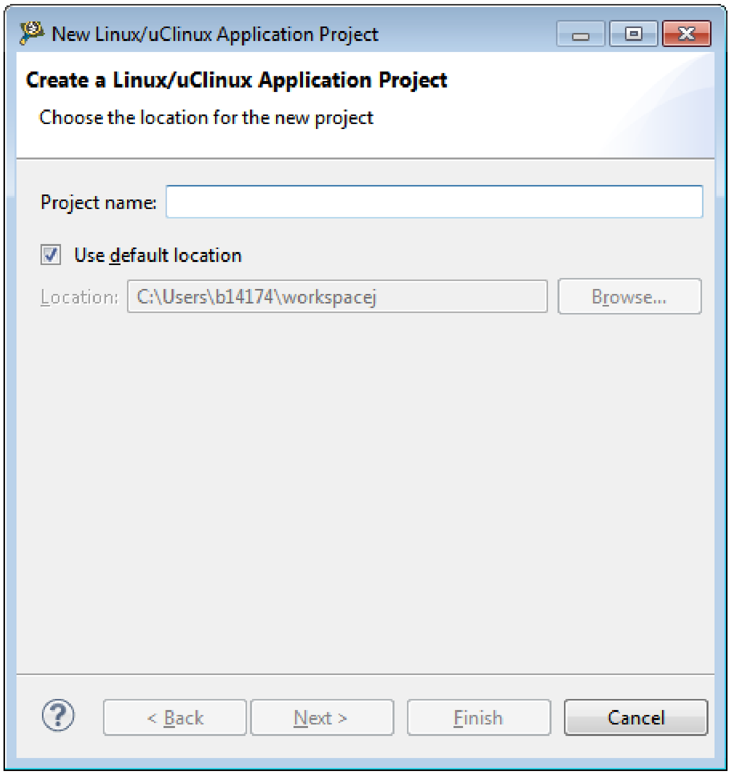
This topic describes the various pages that the wizard displays as it assists you in creating a bareboard project. The pages that the wizard presents can differ based upon the option of project type or execution target.

The pages of the **New Linux/uClinux Application Project** wizard are:

- Create a Linux/uClinux Application Project Page
- Device used for Linux Application Debug page
- Project Language and Output Page
- Connections Page
- Application Debug Options Page

2.3.1 Create a Linux/uClinux Application Project Page

Use this page to specify the project name and the directory where the project files are located.



New Linux/uClinux Application Project

Create a Linux/uClinux Application Project

Choose the location for the new project

Project name:

☑ Use default location

Location: C:\Users\b14174\workspacej

Figure 2-20. Create a Linux/uClinux Application Project page

The table below describes the purpose of the various options.

Table 2-19. Create a Linux/uClinux Application Project Page Settings

Option	Description
Project Name	Enter the name for the new project in this text box.
Use default location	Stores the files required to build the program in the Workbench's current workspace directory. The project files

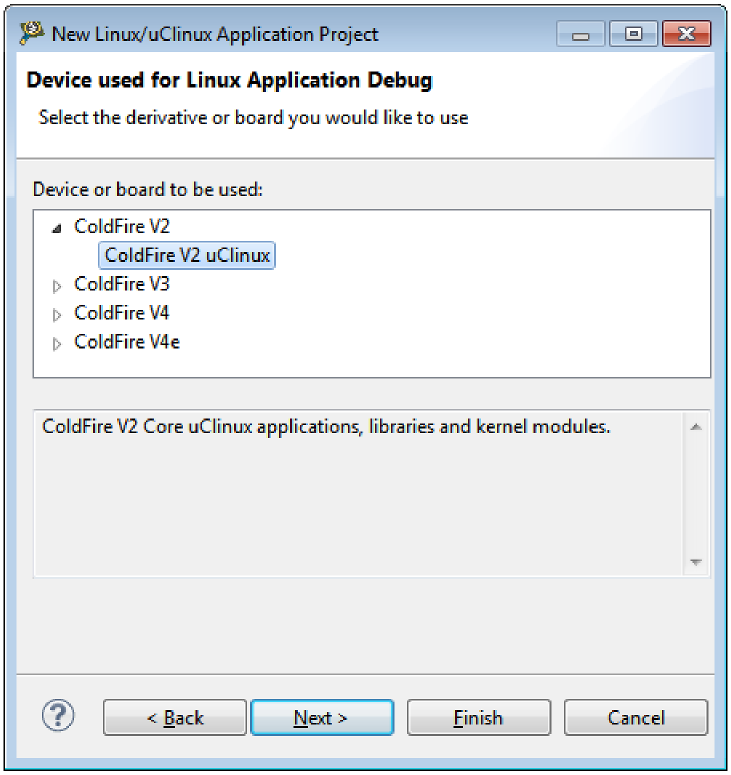
Table continues on the next page...

Table 2-19. Create a Linux/uClinux Application Project Page Settings (continued)

Option	Description
	are stored in the default location. Clear the **Use default location** checkbox and click **Browse** to select a new location.
Location	Specifies the directory that contains the project files. Click **Browse** to navigate to the desired directory. This option is only available when **Use default location** checkbox is clear.

2.3.2 Device used for Linux Application Debug page

Use this page to select the derivative or board you would like to debug.



New Linux/uClinux Application Project
Device used for Linux Application Debug
Select the derivative or board you would like to use
Device or board to be used:
ColdFire V2
ColdFire V2 uClinux
ColdFire V3
ColdFire V4
ColdFire V4e
< Back
Next >
Finish
Cancel

ColdFire V2 Core uClinux applications, libraries and kernel modules.

Figure 2-21. Devices used for Linux Application Debug Page

The table below describes the purpose of the various options.

Table 2-20. Devices used for Linux Application Debug Page

Option	Description
ColdFire V2 > ColdFire V2 uClinux	Select to create ColdFire V2 Core uClinux applications, libraries, and kernel modules.

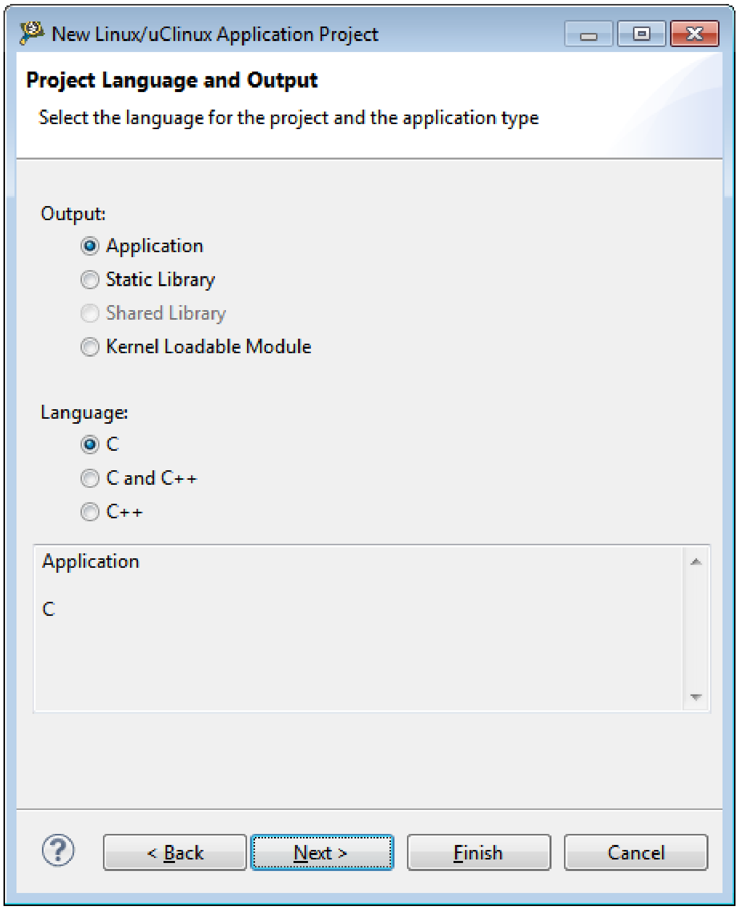
Table continues on the next page...

Table 2-20. Devices used for Linux Application Debug Page (continued)

Option	Description
ColdFire V3 > ColdFire V3 uClinux	Select to create ColdFire V3 Core uClinux applications, libraries, and kernel modules.
ColdFire V4 > ColdFire V4 GNU Linux	Select to create ColdFire V2 Core GNU Linux applications, libraries, and kernel modules.
ColdFire V4e > ColdFire V4e GNU Linux	Select to create ColdFire V4e Core GNU Linux applications, libraries, and kernel modules.

2.3.3 Project Language and Output Page

Use this page to select the programming language that you want to use when writing the program's source code. You can make multiple selections, creating the code in multiple formats.


New Linux/uClinux Application Project

Project Language and Output

Select the language for the project and the application type

Output:

● Application
○ Static Library
○ Shared Library
○ Kernel Loadable Module

Language:

● C
○ C and C++
○ C++

Application

C

Figure 2-22. Project Language and Output Page

The table below describes the purpose of the various options.

NOTE

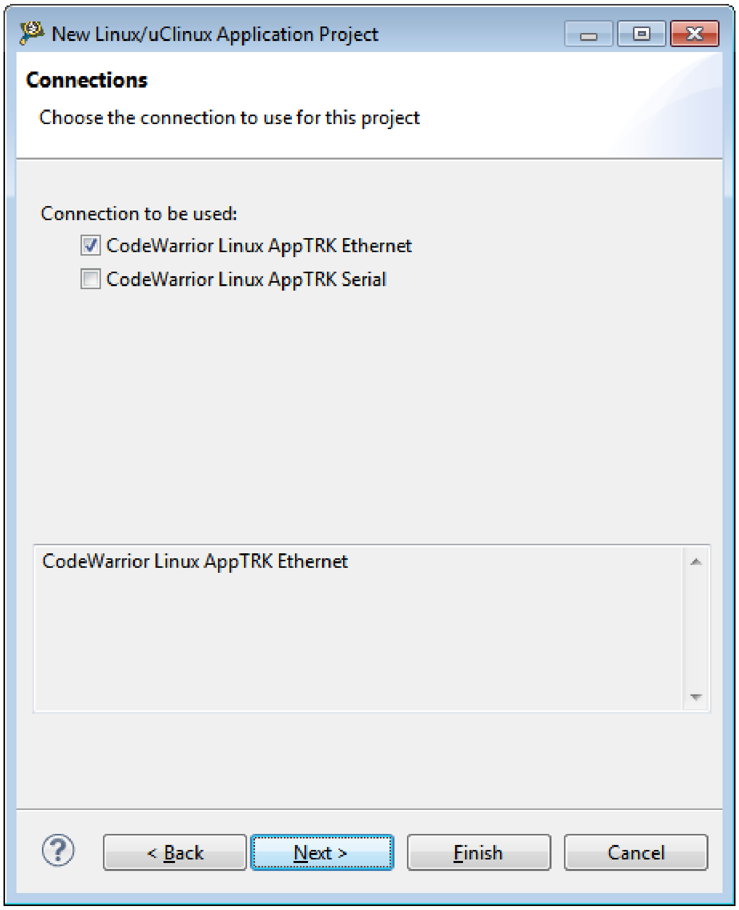
Based on your selection, the IDE may show or hide some options.

Table 2-21. Project Language and Output Page Settings

Option	Description
Application	Select if you want the output to be an application. By default, the extension of a loadable module is `.elf`.
Static Library	Select if you want the output to be a static library. By default, the extension of a static library is `.a`.
Shared Library	Select if you want the output to be a shared library. By default, the extension of a shared library is `.so`.
Kernel Loadable Module	Select if you want the output to be a kernel loadable module. By default, the extension of a loadable module is `.o`.
C	Select to add C language support.
C and C++	Select to add C and C++ language support. Available for Application and Static options only.
C++	Select to add C++ language support. Available for Application and Static options only.

2.3.4 Connections Page

Use this page to select a connection to use for the project. Depending on the selected derivative or board, the connections will appear enabled or grayed out.



New Linux/uClinux Application Project

Connections

Choose the connection to use for this project

Connection to be used:

☑ CodeWarrior Linux AppTRK Ethernet

□ CodeWarrior Linux AppTRK Serial

CodeWarrior Linux AppTRK Ethernet

Figure 2-23. Connections Page

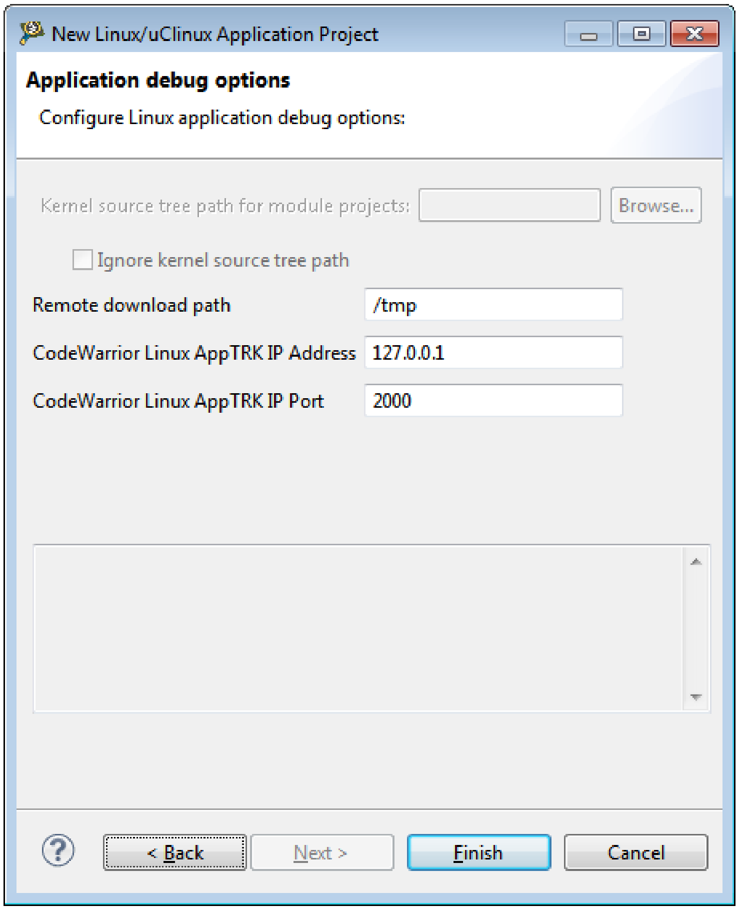
The table below describes the purpose of the various options.

Table 2-22. Connections Page Settings

Option	Description
CodeWarrior Linux AppTRK Ethernet	Available only if the Application option is selected on the Project Language and Output page.
CodeWarrior Linux AppTRK Serial	Available only if the Application option is selected on the Project Language and Output page.

2.3.5 Application Debug Options Page

Use this page to specify the application debug options for a project.



New Linux/uClinux Application Project

Application debug options

Configure Linux application debug options:

Kernel source tree path for module projects:

□ Ignore kernel source tree path

Remote download path: /tmp

CodeWarrior Linux AppTRK IP Address: 127.0.0.1

CodeWarrior Linux AppTRK IP Port: 2000

Figure 2-24. Application Debug Options Page

The table below describes the purpose of the various options.

Table 2-23. Application Debug Options Settings

Option	Description
Kernel source tree path for module projects	Click **Browse** to specify or enter the kernel source tree path for module projects.
Ignore kernel source tree path	Select to ignore the kernel source tree path.
Remote download path	Specify the remote download path.
CodeWarrior Linux AppTRK IP Address	Specify the CodeWarrior Linux AppTRK IP Address.
CodeWarrior Linux AppTRK IP Port	Specify the CodeWarrior Linux AppTRK IP port number.

2.4 Creating Projects

The **New Bareboard Project** and **New Linux/uClinux Application Project** wizards help you to quickly create new projects. The wizard generates a project with placeholder files and default settings (build and launch configurations) specific targets. After the project has been created, you can easily change any default setting to suit your needs.

The following topics explain the steps to create Bareboard and Linux/uClinux Application projects for HCS08, RS08, ColdFire V1, ColdFire V2-4e, Kinetis, Qorivva and S12Z derivatives.

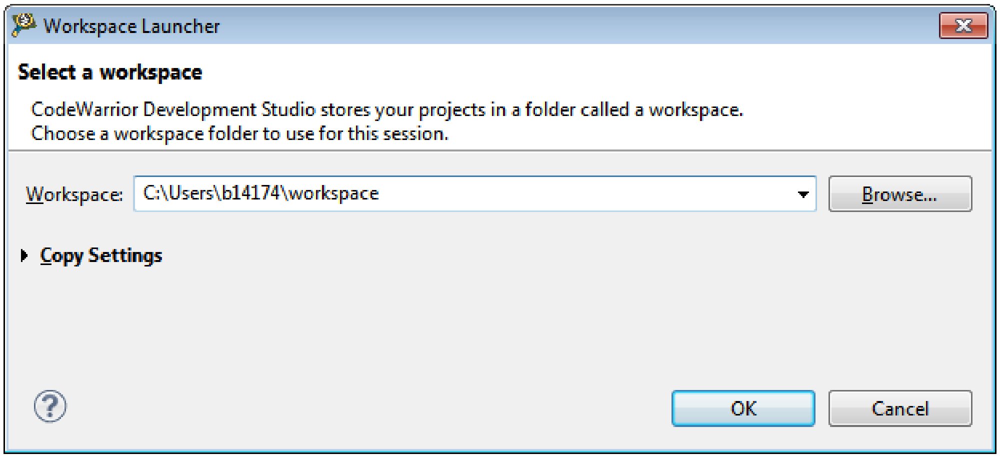
- Launching Workbench
- Creating Bareboard Projects
- Creating Linux/uClinux Application Project

2.4.1 Launching Workbench

To launch the CodeWarrior IDE for creating, building and debugging projects:

1. Select **Start > Programs > Freescale CodeWarrior > CW for MCU v10.x > CodeWarrior**.

   The **WorkSpace Launcher** dialog box appears and prompts you to select a workspace to use.



Workspace Launcher

Select a workspace

CodeWarrior Development Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\b14174\workspace

Figure 2-25. WorkSpace Launcher Dialog Box

2. Click **OK** to accept the default workspace. To use a workspace different from the default, click **Browse** and specify the desired workspace.

   The IDE starts and displays the **Welcome** page.

NOTE

You can also select the **Use this as the default and do not ask again** checkbox to set default/selected path as a default location for storing all your projects.

3. Click the **Go to Workbench** link.

   The **Workbench** window appears.

2.4.2 Creating Bareboard Projects

The following topics explain the steps to create bareboard projects for S08/RS08, ColdFire, Kinetis, Qorivva, 56800/E(DSC), and S12Z architectures.

- Creating Target Board Projects for S08/RS08
- Creating Target Board Projects for ColdFire V1/ColdFire+ V1/Sensors
- Creating Target Board Projects for ColdFire V2/V3/V4/V4e/Vx
- Creating Target Board Project for Kinetis
- Creating Target Board Project for Qorivva
- Creating Target Board Projects for 56800/E (DSC)
- Creating Target Board Projects for S12Z

NOTE

The Full Chip Simulation is supported only by the S08, RS08 and 56800/E (DSC) derivatives.

2.4.2.1 Creating Target Board Projects for S08/RS08

To create a new simulator project for an S08/RS08 derivative using the **New Bareboard Project** wizard:

1. Launch the Workbench.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

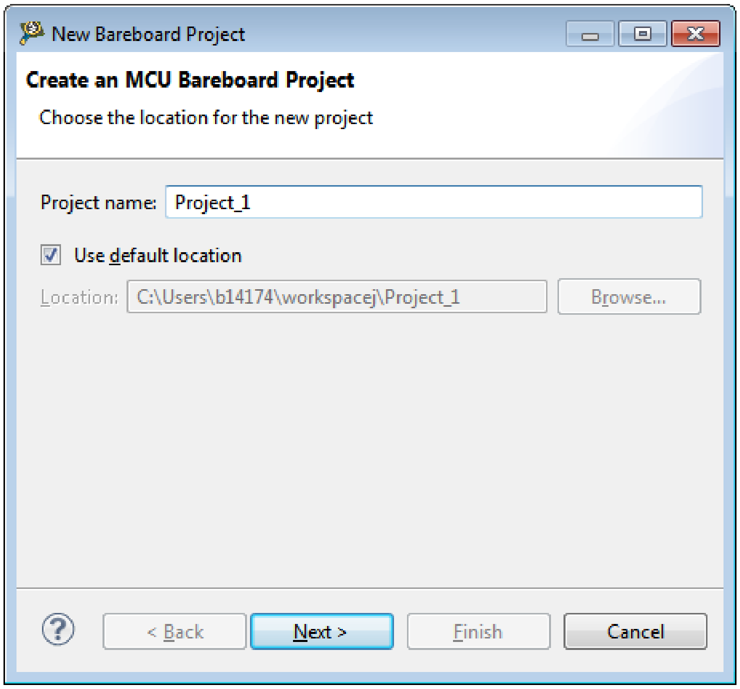
2. Select **File > New > Bareboard Project** , from the IDE menu bar.

The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `Project_1`.

NOTE

If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name: Project_1

☑ Use default location

Location: C:\Users\b14174\workspacej\Project_1

Figure 2-26. New Bareboard Project wizard - Create an MCU Bareboard Project page

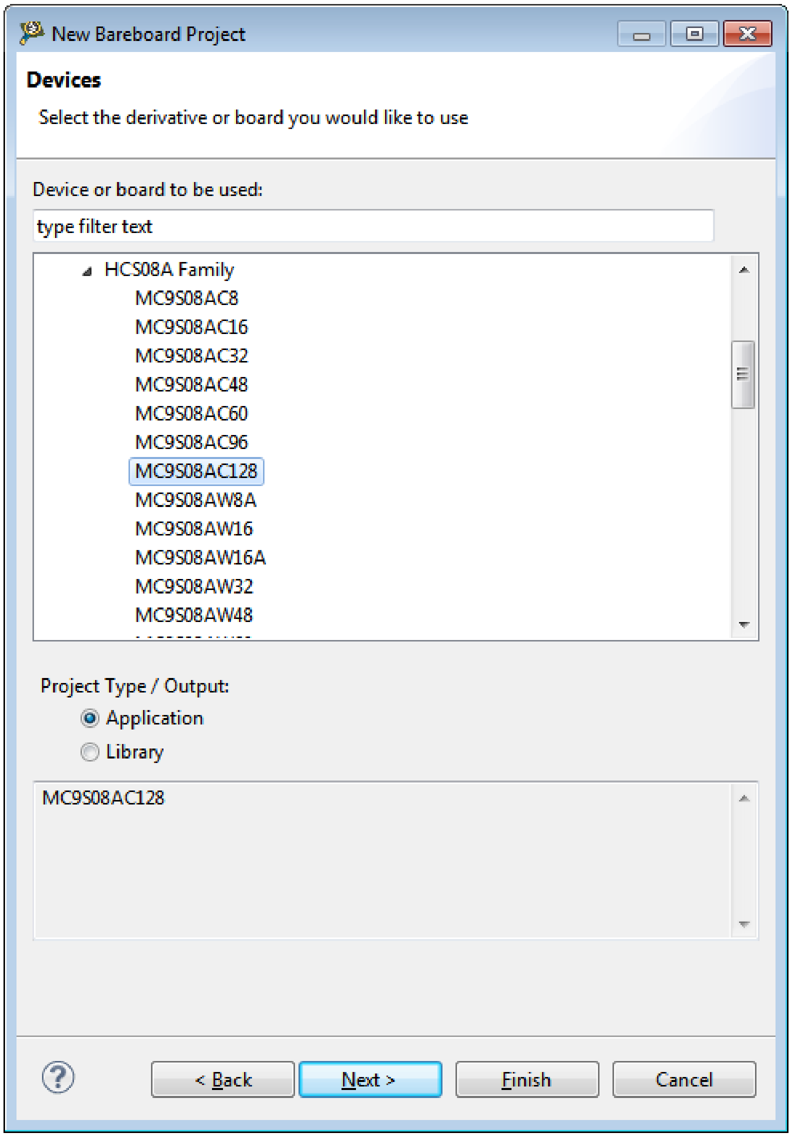
4. Click **Next**.

The **Devices** page appears.

5. Expand the desired tree control and select the derivative or board you would like to use. For example, select **S08 > HCS08A Family > MC9S08AC128** for an S08 derivative.

NOTE

If you want create a library project, select **Library** from the **Project Type / Output** group. For this example keep the default setting to create an application project.



Device or board to be used:

type filter text

- HCS08A Family
  - MC9S08AC8
  - MC9S08AC16
  - MC9S08AC32
  - MC9S08AC48
  - MC9S08AC60
  - MC9S08AC96
  - MC9S08AC128
  - MC9S08AW8A
  - MC9S08AW16
  - MC9S08AW16A
  - MC9S08AW32
  - MC9S08AW48

Project Type / Output:

● Application

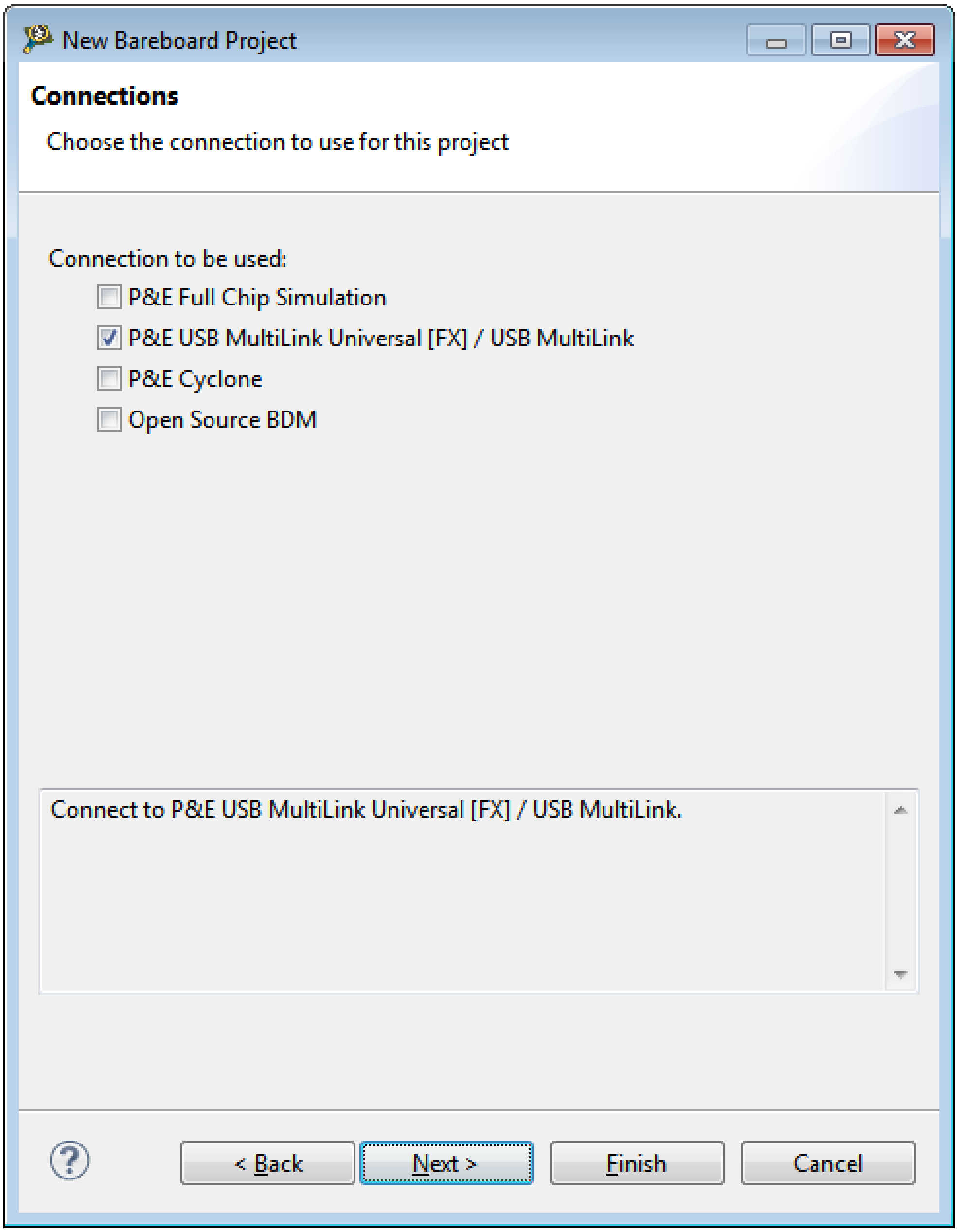
○ Library

MC9S08AC128

Figure 2-27. New Bareboard Project wizard - Devices page

6. Click **Next**.

   The **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- □ P&E Full Chip Simulation
- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ Open Source BDM

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-28. New Bareboard Project wizard - Connections Page

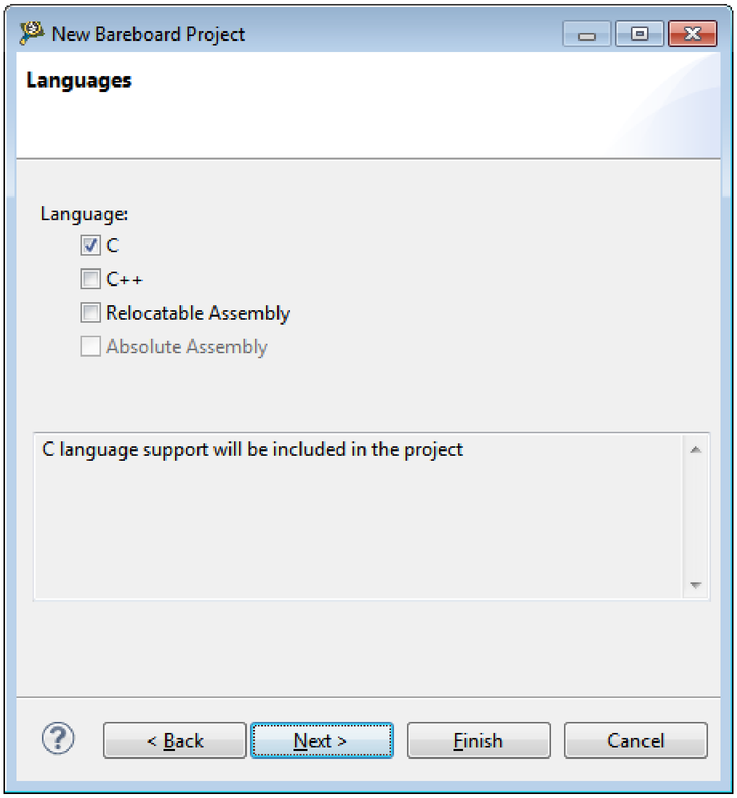
7. Check the appropriate connection.

NOTE

You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink Universal [FX] / USB MultiLink** checkbox is checked.

8. Click **Next**.

   The **Languages** page appears.



New Bareboard Project

Languages

Language:

☑ C

□ C++

□ Relocatable Assembly

□ Absolute Assembly

C language support will be included in the project

Figure 2-29. New Bareboard Project wizard - Languages page

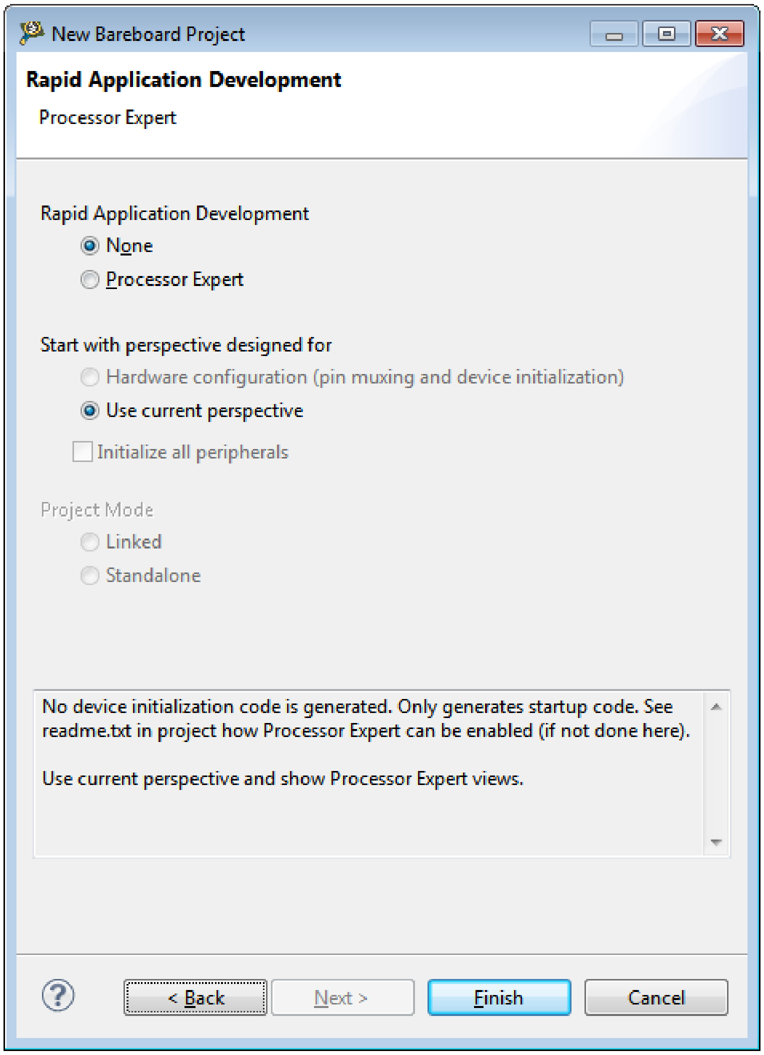
9. Select the programming language you want to use. For example, check the **C** checkbox.

NOTE

Availability of the options depends on the derivative or board you selected. For example, **C++** option appears grey and is not accessible for RS08 derivatives.

10. Click **Next**.

    The **Rapid Application Development** page appears.



New Bareboard Project

Rapid Application Development

Processor Expert

Rapid Application Development

● None

○ Processor Expert

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

Project Mode

○ Linked

○ Standalone

No device initialization code is generated. Only generates startup code. See readme.txt in project how Processor Expert can be enabled (if not done here).

Use current perspective and show Processor Expert views.

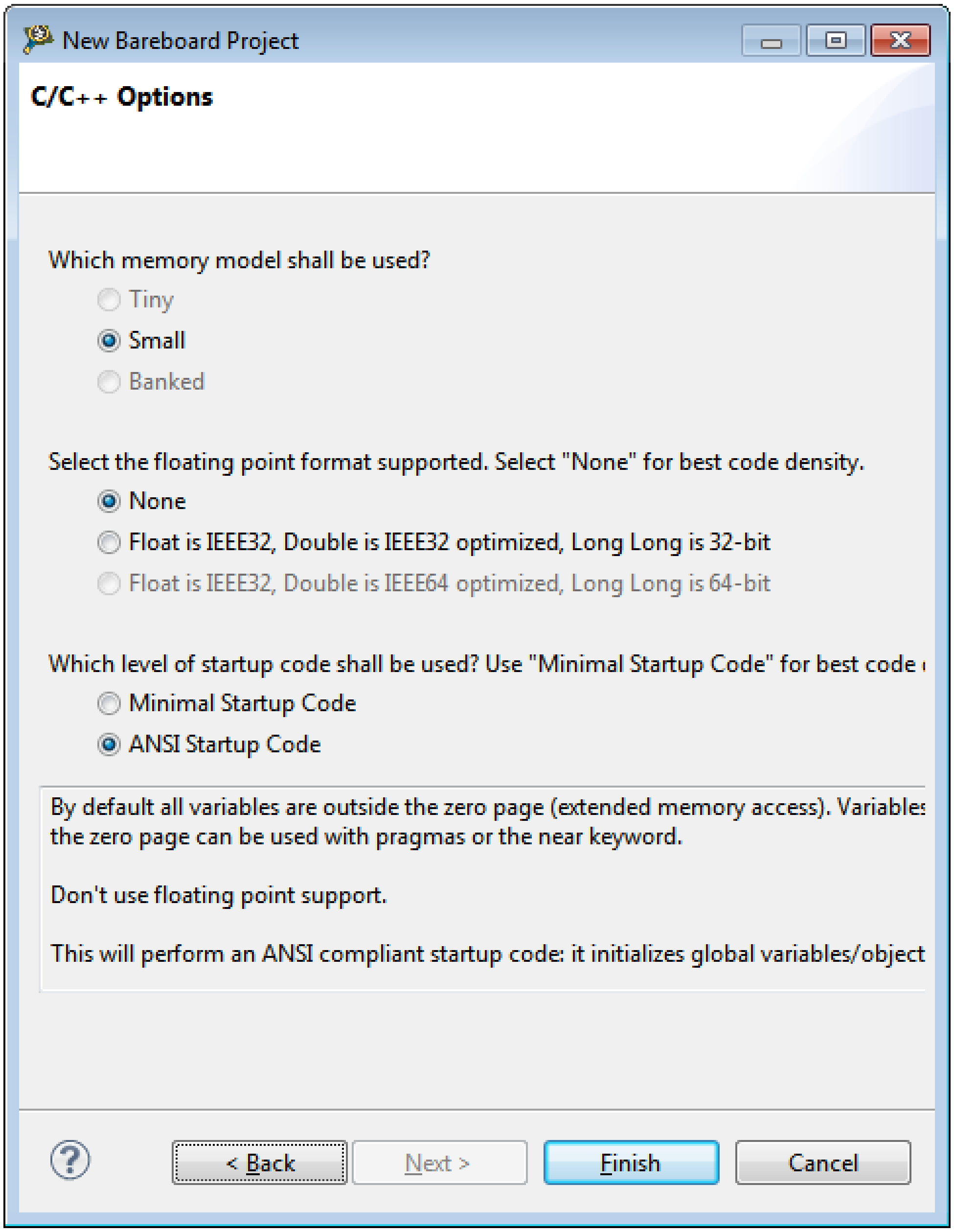
Figure 2-30. New Bareboard Project wizard - Rapid Application Development page

NOTE

The **Processor Expert** option is disabled, if you select only the **Relocatable Assembly** or **Absolute Assembly** in the **Connections** page.

11. Select the appropriate option to support rapid application development.
    - **None** - Select to generate only startup code.
    - **Processor Expert** - Select to generate the device initialization code, including low-level drivers.
12. Click **Next**.

    The **C/C++ Options** page appears.



New Bareboard Project

C/C++ Options

Which memory model shall be used?

○ Tiny
● Small
○ Banked

Select the floating point format supported. Select "None" for best code density.

● None
○ Float is IEEE32, Double is IEEE32 optimized, Long Long is 32-bit
○ Float is IEEE32, Double is IEEE64 optimized, Long Long is 64-bit

Which level of startup code shall be used? Use "Minimal Startup Code" for best code

○ Minimal Startup Code
● ANSI Startup Code

By default all variables are outside the zero page (extended memory access). Variables
the zero page can be used with pragmas or the near keyword.

Don't use floating point support.

This will perform an ANSI compliant startup code: it initializes global variables/object

Figure 2-31. New Bareboard Project wizard - C/C++ Options page

NOTE

> Some options may appear grey and are not accessible. The availability of the options depends on the derivative or board you selected.

13. Select the appropriate level of startup code, memory model, and floating point format.
14. Click **Finish**.

The wizard creates a new simulator project for the HCS08 architecture according to your specifications. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

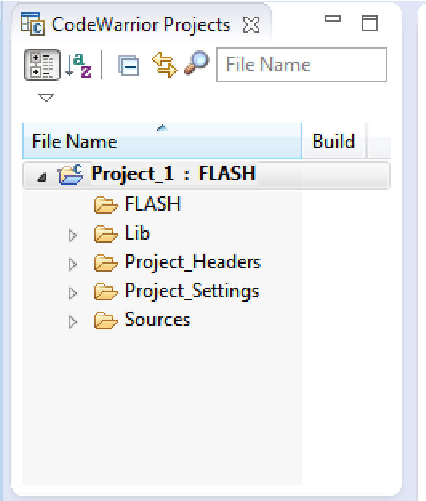


Figure 2-32. CodeWarrior Projects View

The new project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

NOTE

You can click **Finish** at any step in the Project Wizard to save the project with the default settings.

2.4.2.2 Creating Target Board Projects for ColdFire V1/ColdFire+ V1/ Sensors

To create a new target board project for a ColdFire V1/ColdFire+ V1/Sensors derivative using the **New Bareboard Project** wizard**:**

1. Launch the Workbench.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

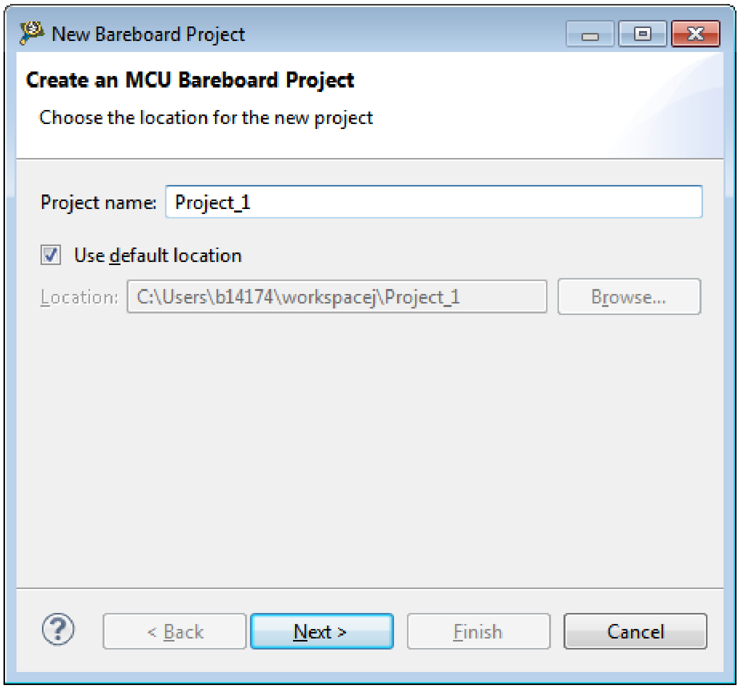
2. Select **File > New > Bareboard Project** , from the IDE menu bar.

   The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `TargetProject_1`.

NOTE

If you do not want to use the default location, clear the Use default location checkbox. In the Location text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click Browse and select the desired location from the Browse For Folder dialog box and click OK. Ensure that you append the path with the name of the project to create a new location for your project.


New Bareboard Project
Create an MCU Bareboard Project
Choose the location for the new project
< Back
Next >
Finish
Cancel
Browse...

Project name: Project_1

☑ Use default location

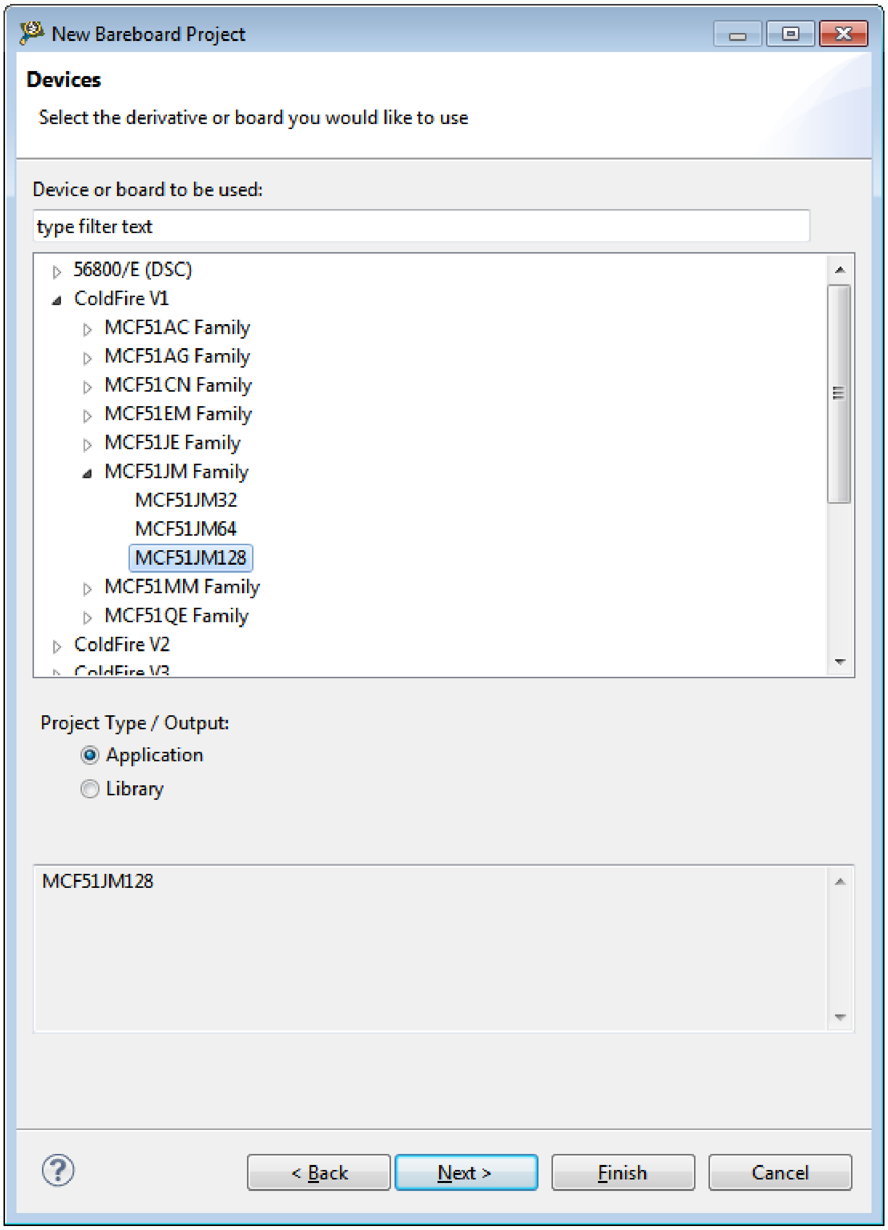
Location: C:\Users\b14174\workspacej\Project_1

Figure 2-33. New Bareboard Project wizard - Create an MCU Bareboard Project Page

4. Click **Next**.

   The **Devices** page appears.

5. Expand the tree control and select the derivative or board you would like to use. For example, select **ColdFire V1 > MCF51JM Family > MCF51JM128**.



New Bareboard Project

Devices

Select the derivative or board you would like to use

Device or board to be used:

type filter text

- 56800/E (DSC)
- ColdFire V1
  - MCF51AC Family
  - MCF51AG Family
  - MCF51CN Family
  - MCF51EM Family
  - MCF51JE Family
  - MCF51JM Family
    - MCF51JM32
    - MCF51JM64
    - MCF51JM128
  - MCF51MM Family
  - MCF51QE Family
- ColdFire V2
- ColdFire V3

Project Type / Output:

● Application

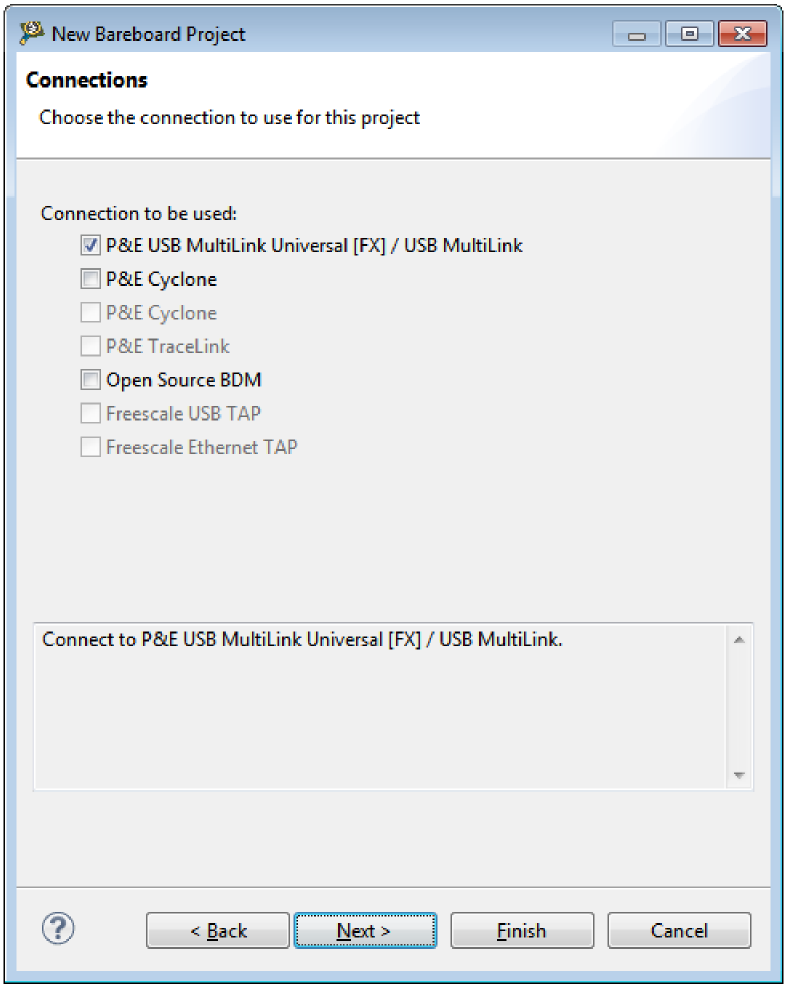
○ Library

MCF51JM128

Figure 2-34. New Bareboard Project Wizard - Devices Page

6. Click **Next**.

   The **Connections** page appears.



New Bareboard Project

Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ P&E Cyclone
- □ P&E TraceLink
- □ Open Source BDM
- □ Freescale USB TAP
- □ Freescale Ethernet TAP

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-35. New Bareboard Project Wizard - Connections Page

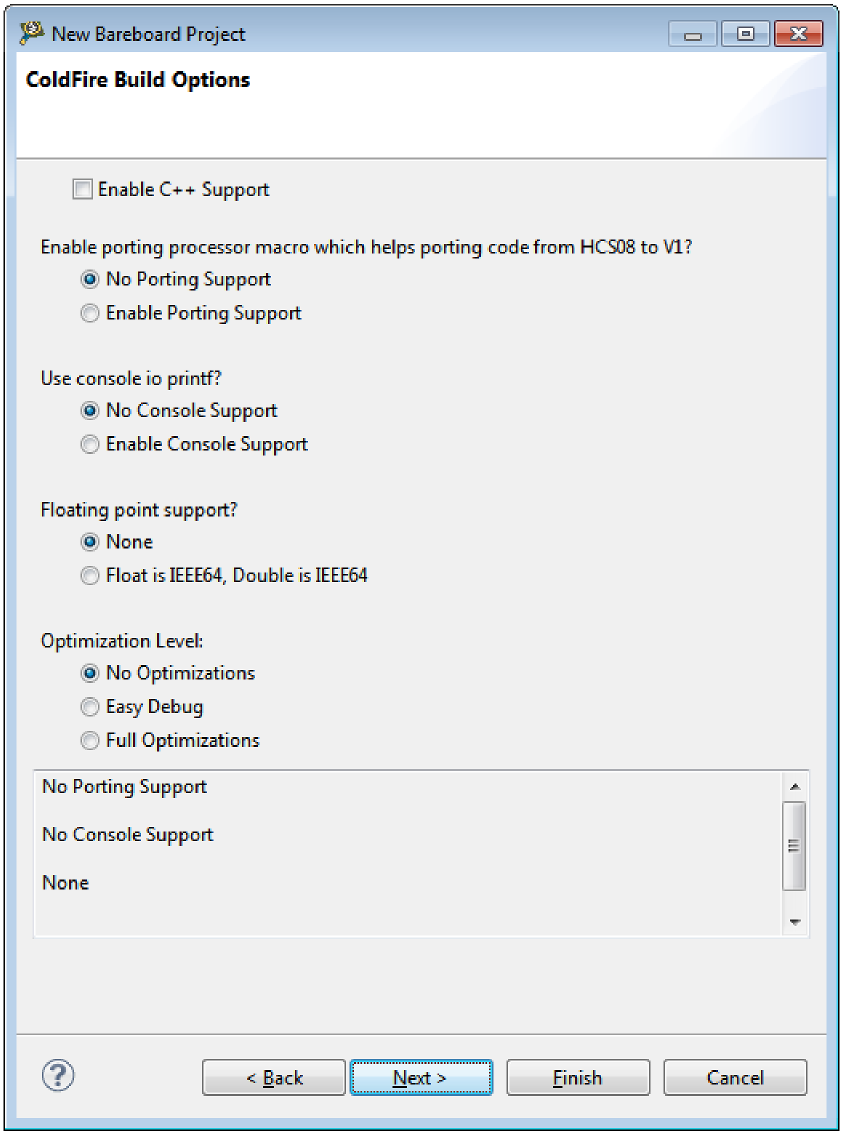
7. Check the appropriate connection.

NOTE

> You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink Universal [FX] / USB MultiLink** checkbox is checked.

8. Click **Next**.

   The **ColdFire Build Options** page appears.



New Bareboard Project

ColdFire Build Options

□ Enable C++ Support

Enable porting processor macro which helps porting code from HCS08 to V1?

● No Porting Support

○ Enable Porting Support

Use console io printf?

● No Console Support

○ Enable Console Support

Floating point support?

● None

○ Float is IEEE64, Double is IEEE64

Optimization Level:

● No Optimizations

○ Easy Debug

○ Full Optimizations

No Porting Support

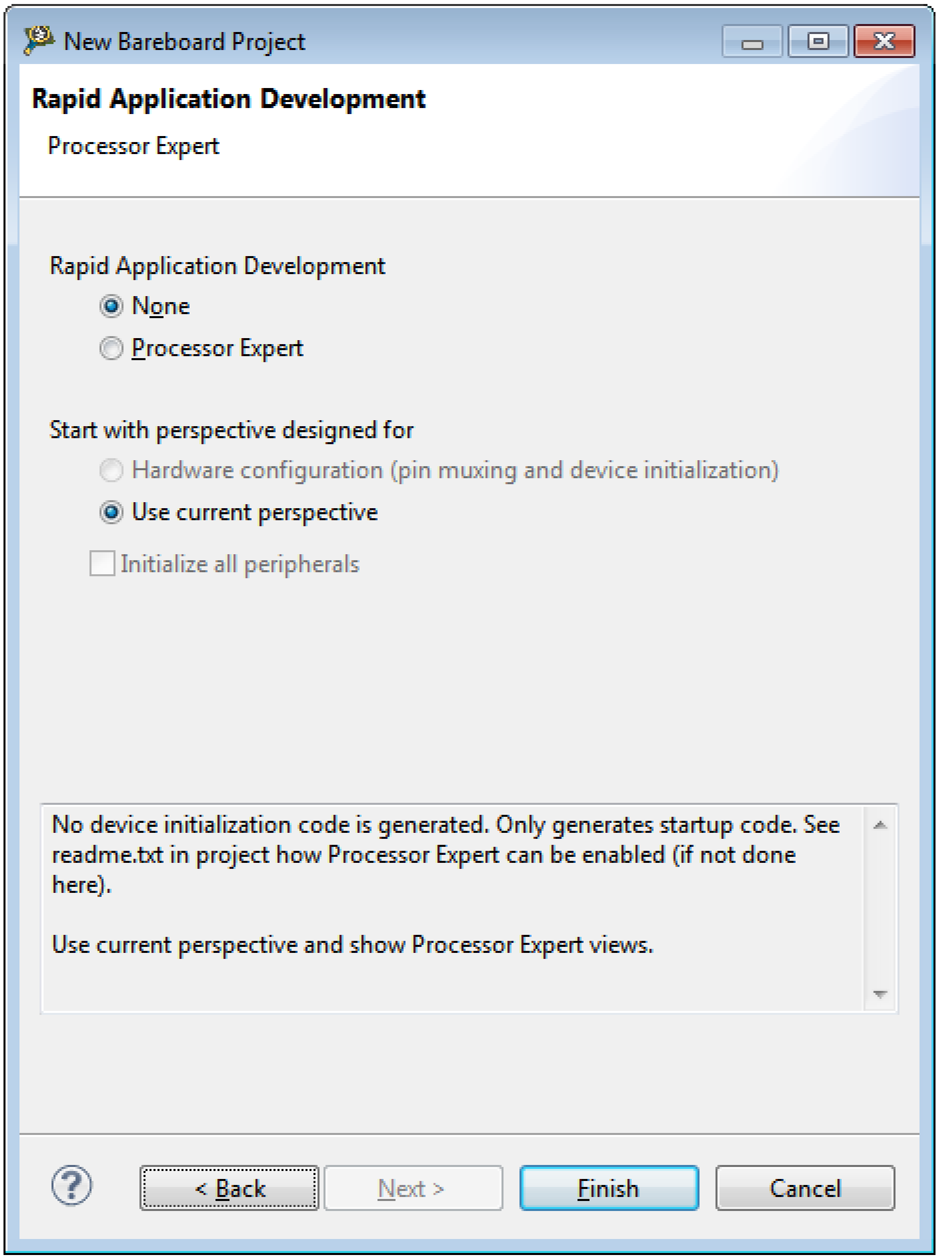
No Console Support

None

Figure 2-36. New Bareboard Project Wizard - ColdFire Build Options Page

9. Select the appropriate options to enable C++, porting processor macro, console, floating point supports and optimization level.
10. Click **Next**.

    The **Rapid Application Development** page appears.



Rapid Application Development

Processor Expert

Rapid Application Development

● None

○ Processor Expert

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

No device initialization code is generated. Only generates startup code. See readme.txt in project how Processor Expert can be enabled (if not done here).

Use current perspective and show Processor Expert views.

Figure 2-37. New Bareboard Project Wizard - Rapid Application Development Page

NOTE

The **Processor Expert** options is disabled, if you enables the C++ support in the **ColdFire Build Options** page.

11. Select the appropriate option to support rapid application development.
12. Click **Finish**.

    The wizard creates a new target board project for the ColdFire V1 architecture. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

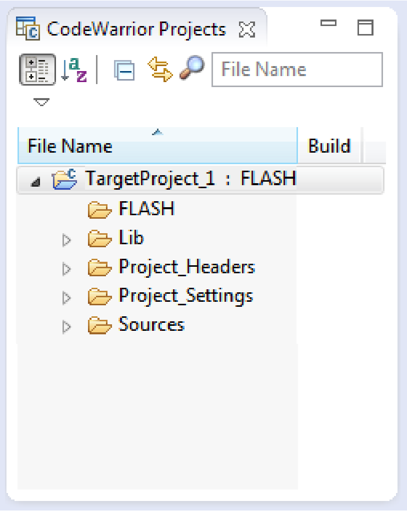


Figure 2-38. CodeWarrior Projects View

The new project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

NOTE

You can click **Finish** at any step in the Project Wizard to save the project with the default settings.

2.4.2.3 Creating Target Board Projects for ColdFire V2/V3/V4/V4e/Vx

To create a new target board project for a ColdFire V2/V3/V4/V4e/Vx derivative using the **New Bareboard Project** wizard:

1. Launch the Workbench.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

2. Select **File > New > Bareboard Project** , from the IDE menu bar.

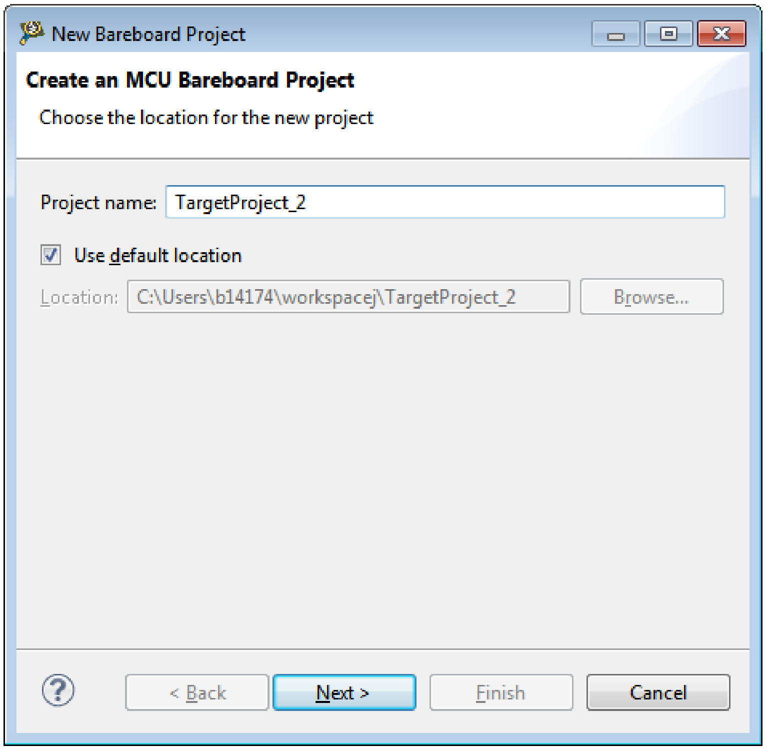
   The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `TargetProject_2`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter

the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name: TargetProject_2

☑ Use default location

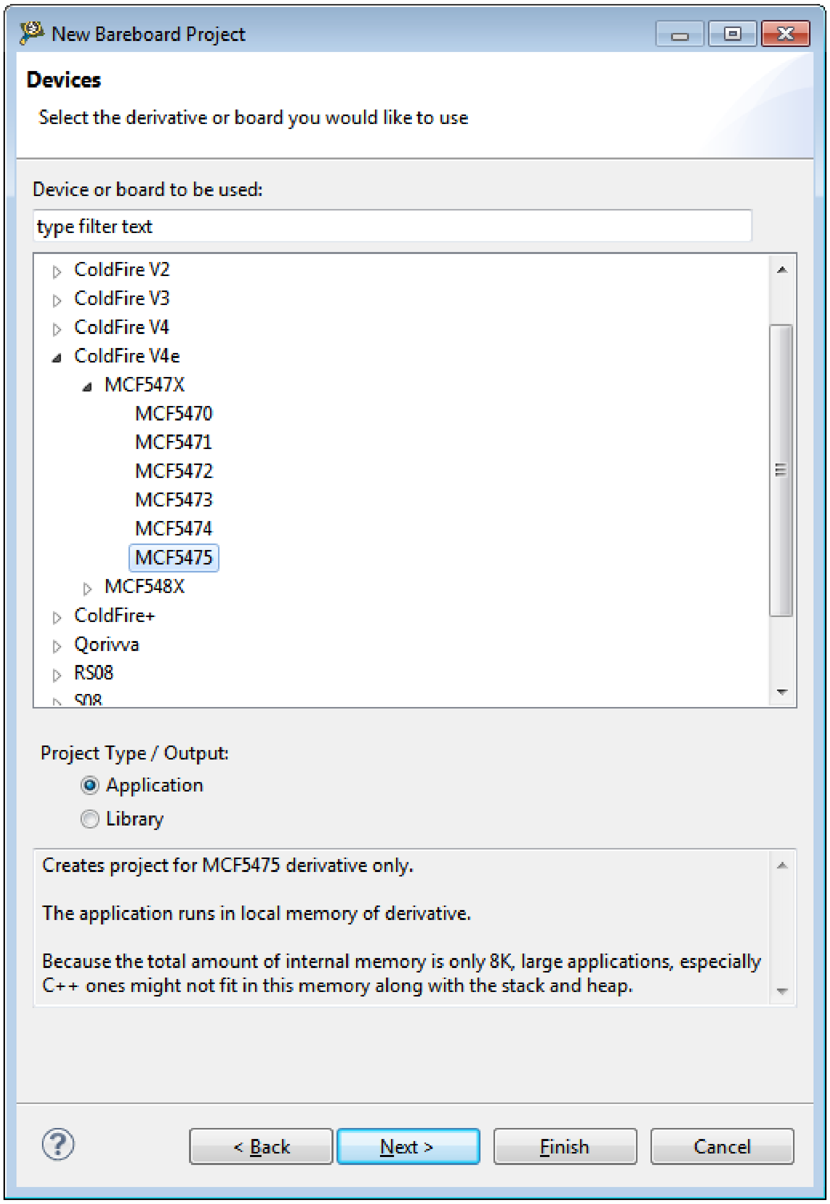
Location: C:\Users\b14174\workspacej\TargetProject_2

Figure 2-39. New Bareboard Project Wizard - Create an MCU Bareboard Project Page

4. Click **Next**.

   The **Devices** page appears.

5. Expand the desired tree control and select the derivative or board you would like to use. For example, select **ColdFire V4e > MCF547X > MCF5475**.



Creates project for MCF5475 derivative only.

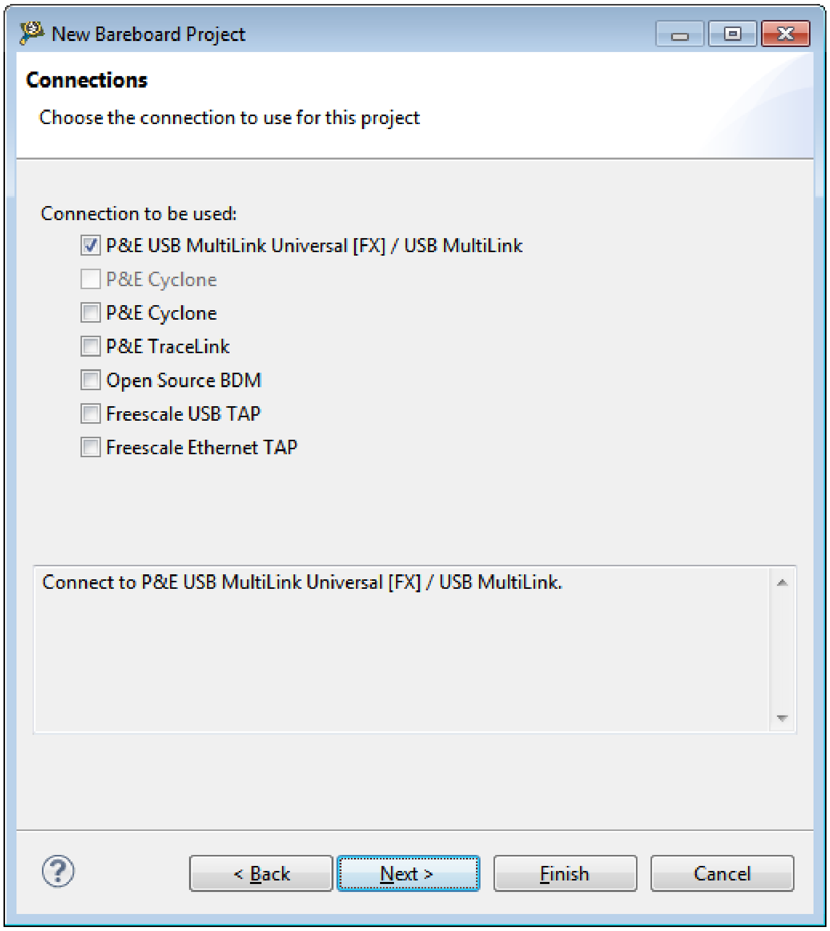
The application runs in local memory of derivative.

Because the total amount of internal memory is only 8K, large applications, especially C++ ones might not fit in this memory along with the stack and heap.

Figure 2-40. New Bareboard Project Wizard - Devices Page

6. Click **Next**.

   The **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- ☐ P&E Cyclone
- ☐ P&E Cyclone
- ☐ P&E TraceLink
- ☐ Open Source BDM
- ☐ Freescale USB TAP
- ☐ Freescale Ethernet TAP

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-41. New Bareboard Project Wizard - Connections Page

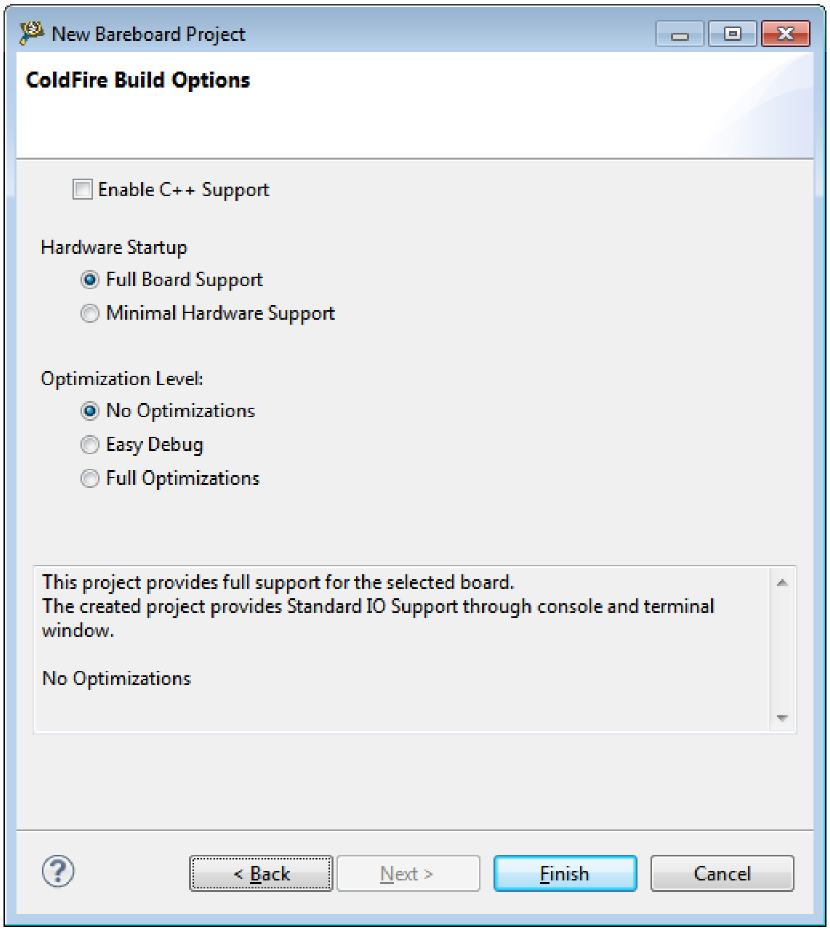
7. Check the appropriate connection.

NOTE

> You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink Universal [FX] / USB MultiLink** checkbox is checked.

8. Click **Next**.

   The **ColdFire Build Options** page appears.


New Bareboard Project

ColdFire Build Options

□ Enable C++ Support

Hardware Startup

● Full Board Support

○ Minimal Hardware Support

Optimization Level:

● No Optimizations

○ Easy Debug

○ Full Optimizations

This project provides full support for the selected board.
The created project provides Standard IO Support through console and terminal window.

No Optimizations

Figure 2-42. New Bareboard Project Wizard - ColdFire Build Options Page

9. Select the appropriate options to enable C++ support, hardware startup level, and the optimization level for your project.
10. Click **Finish**.

    The wizard creates a new target board project for the ColdFire V4e architecture. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

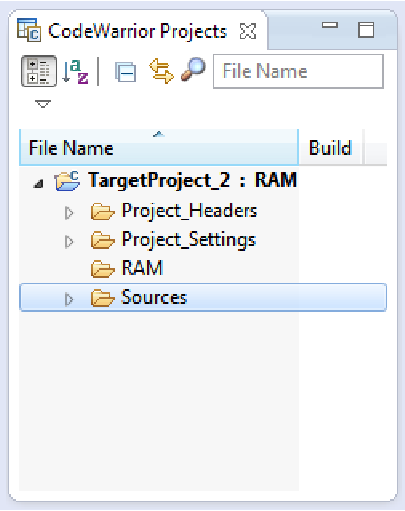


Figure 2-43. CodeWarrior Projects View

The new project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

NOTE

You can click **Finish** at any step in the Project Wizard to save the project with the default settings.

2.4.2.4 Creating Target Board Project for Kinetis

To create a new target board project for a Kinetis derivative using the **New Bareboard Project** wizard:

1. Launch the Workbench.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

2. Select **File > New > Bareboard Project**, from the IDE menu bar.

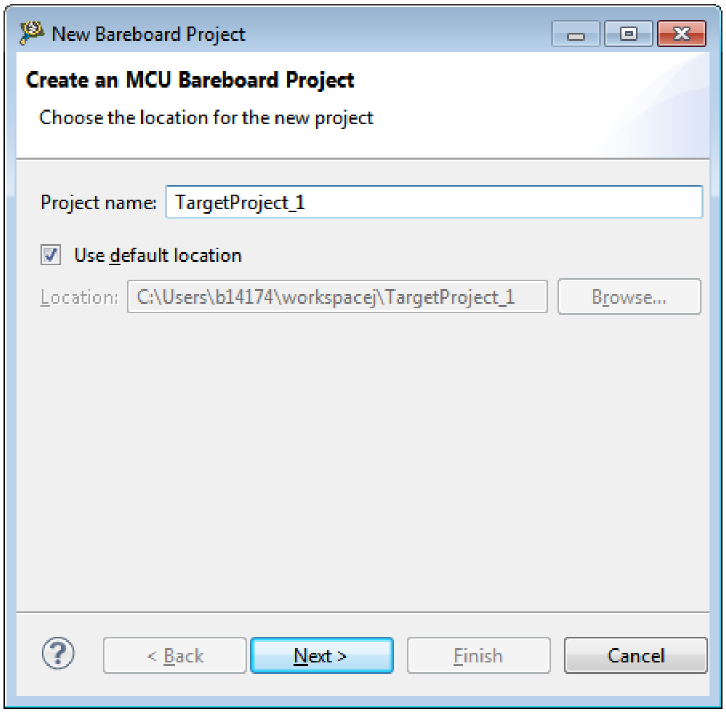
   The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `TargetProject_1`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter

the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name: TargetProject_1

☑ Use default location

Location: C:\Users\b14174\workspacej\TargetProject_1

Figure 2-44. New Bareboard Project Wizard - Create an MCU Bareboard Project Page

4. Click **Next**.

   The **Devices** page appears.

5. Expand the desired tree control and select the derivative or board you would like to use. For example, select **Kinetis K Series > K1x Family > K10D (100 MHz) Family > MK10DN512Z**.

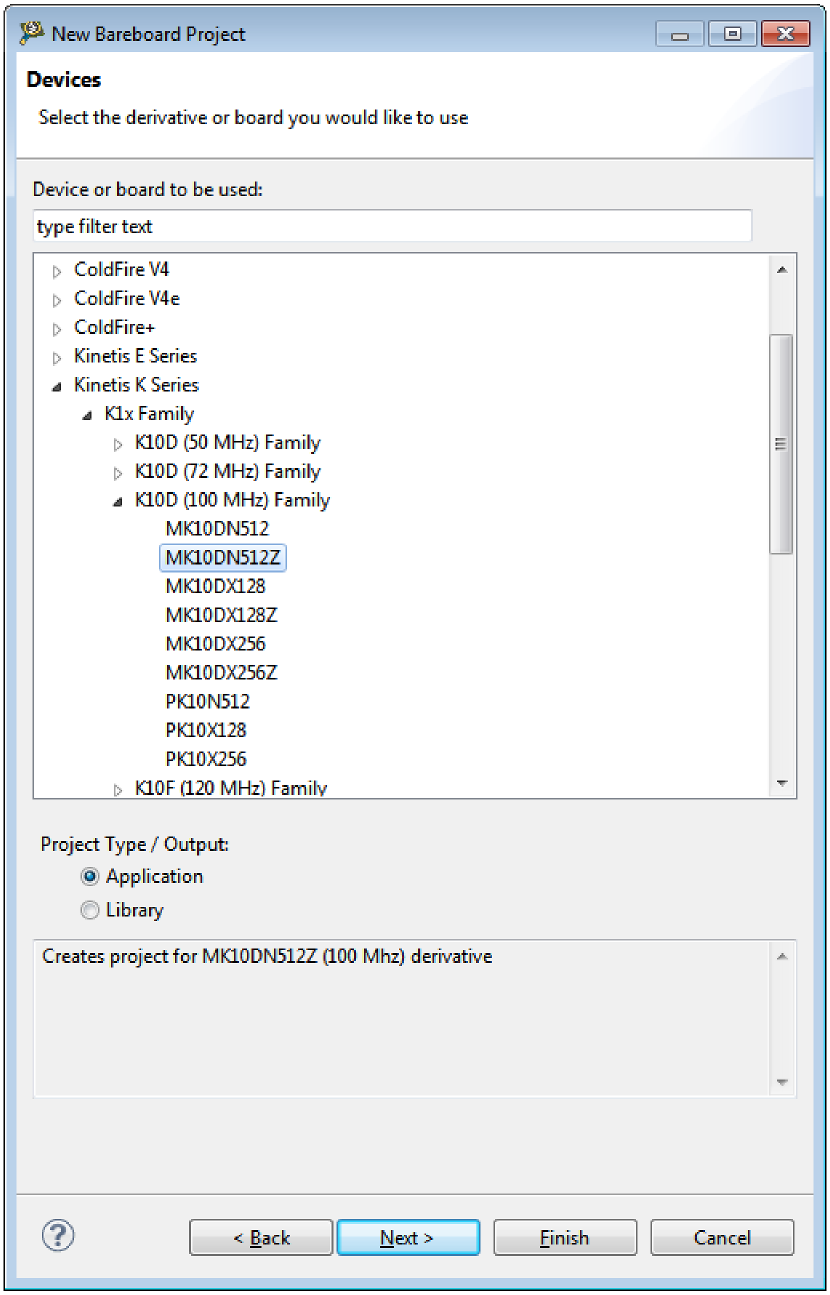


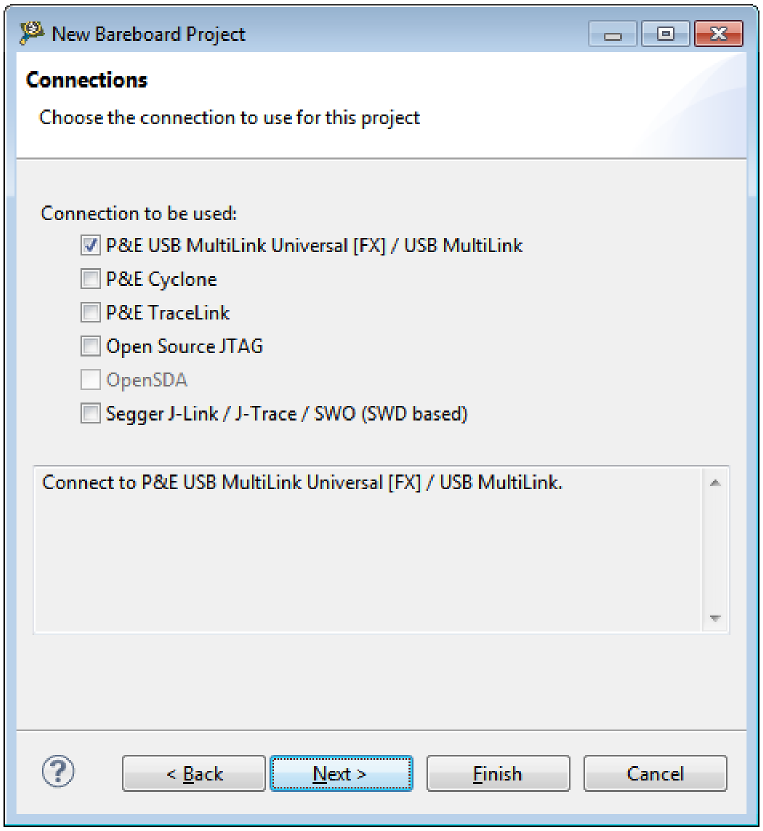
Figure 2-45. New Bareboard Project Wizard - Devices Page

NOTE

A new Kinetis family, named L-Family (or L-Series) is introduced. All the processors within this family have a Cortex-M0+ core, also referred as Flycatcher (an ARM P0 (48 Mhz device), having M0+ core).

6. Click **Next** .

   The **Connections** page appears.



New Bareboard Project

Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- ☐ P&E Cyclone
- ☐ P&E TraceLink
- ☐ Open Source JTAG
- ☐ OpenSDA
- ☐ Segger J-Link / J-Trace / SWO (SWD based)

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-46. New Bareboard Project Wizard - Connections Page

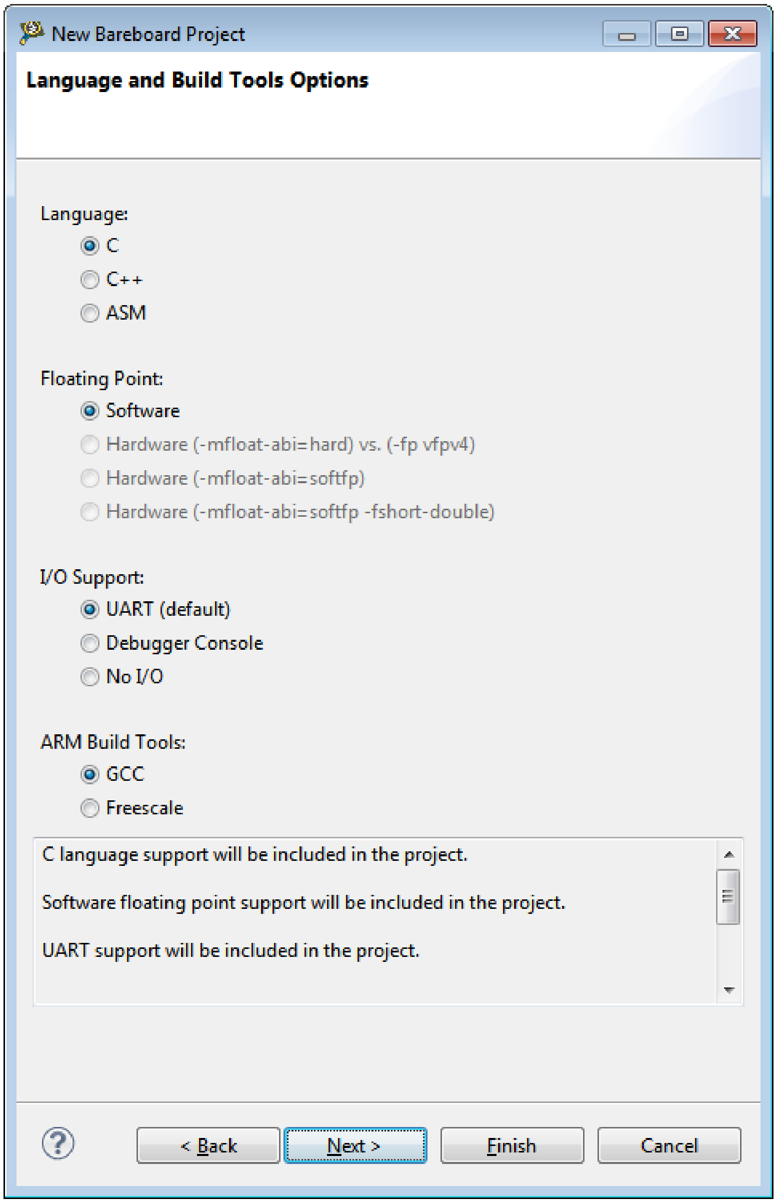
7. Check the appropriate connection.

NOTE

> You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink Universal [FX] / USB MultiLink** checkbox is checked.

8. Click **Next**.

   The **Language and Build Tools Options** page appears.



Language and Build Tools Options

Language:

● C

○ C++

○ ASM

Floating Point:

● Software

○ Hardware (-mfloat-abi=hard) vs. (-fp vfpv4)

○ Hardware (-mfloat-abi=softfp)

○ Hardware (-mfloat-abi=softfp -fshort-double)

I/O Support:

● UART (default)

○ Debugger Console

○ No I/O

ARM Build Tools:

● GCC

○ Freescale

C language support will be included in the project.

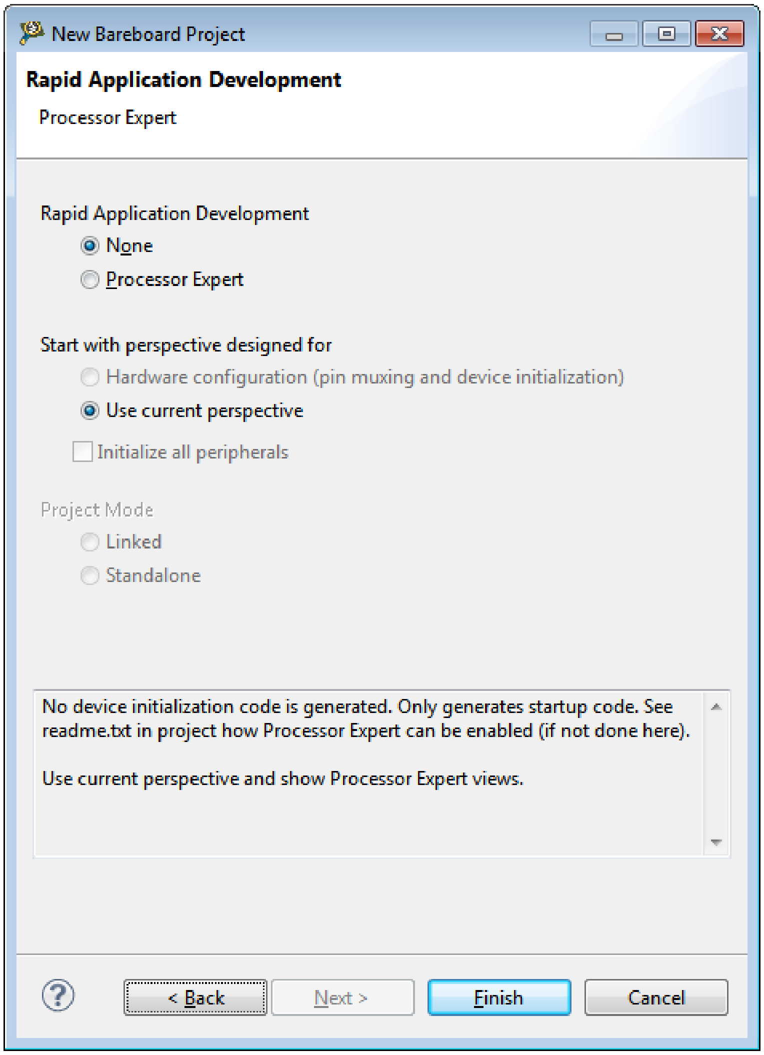
Software floating point support will be included in the project.

UART support will be included in the project.

Figure 2-47. New Bareboard Project Wizard - Languages Page

9. Select the appropriate options to enable C++, porting processor macro, console, and floating point supports.
10. Click **Next**.

    The **Rapid Application Development** page appears.



Rapid Application Development

● None

○ Processor Expert

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

Project Mode

○ Linked

○ Standalone

No device initialization code is generated. Only generates startup code. See readme.txt in project how Processor Expert can be enabled (if not done here).

Use current perspective and show Processor Expert views.

Figure 2-48. New Bareboard Project Wizard - Rapid Application Development Page

11. Select the appropriate option to support rapid application development.
    - **None** - Select to generate only startup code.
    - **Processor Expert** - Select to generate the device initialization code, including low-level drivers.

NOTE

If you select the **Processor Expert** option, clicking the Next button will display the **Processor Expert MCU Pin Variants and Configuration** page. Here you can select the required microcontroller variant and its configuration.

12. By default the project created will appear and start in the current perspective.
13. Click **Finish**.

The wizard creates the target board project for the Kinetis architecture. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

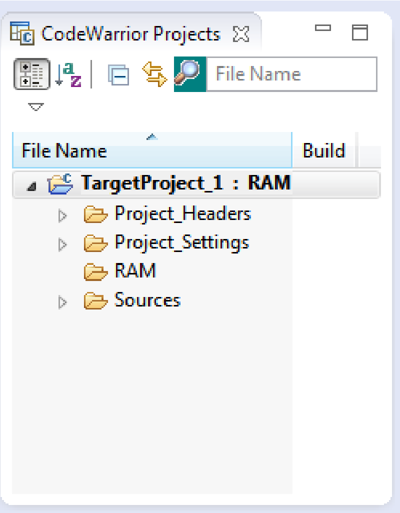


Figure 2-49. CodeWarrior Projects View

2.4.2.5 Creating Target Board Project for Qorivva

To create a new target board project for a Qorivva derivative using the **New Bareboard Project** wizard:

1. Launch the Workbench.

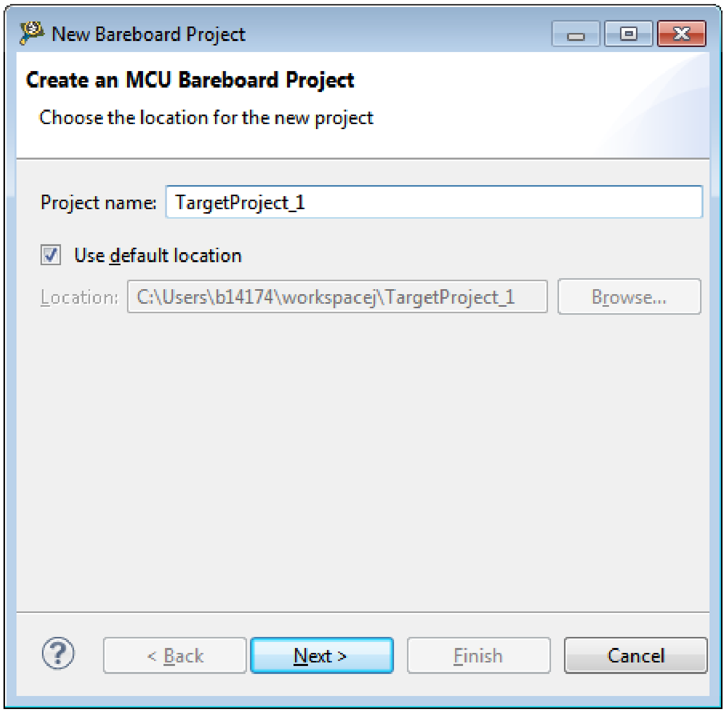
   NOTE
   For information about launching the Workbench, refer to the topic Launching Workbench.

2. Select **File > New > Bareboard Project**, from the IDE menu bar.

   The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `TargetProject_1`.

   NOTE
   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name: TargetProject_1

☑ Use default location

Location: C:\Users\b14174\workspacej\TargetProject_1

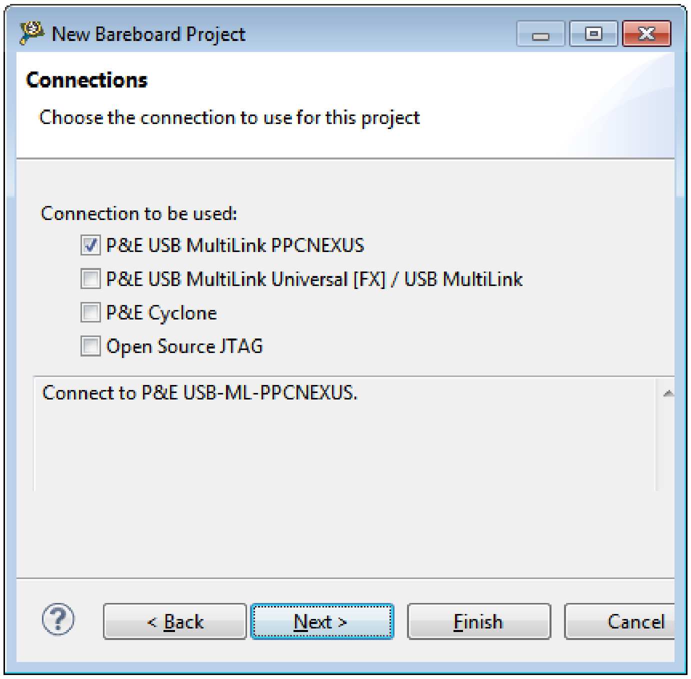
Figure 2-50. New Bareboard Project Wizard - Create an MCU Bareboard Project Page

4. Click **Next**.

   The **Devices** page appears.

5. Expand the tree control and select the derivative or board you would like to use. For example, select **Qorivva > MPC56xxK Family > MPC5673K** for Qorivva derivatives.
6. Click **Next**.

   The **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

☑ P&E USB MultiLink PPCNEXUS
□ P&E USB MultiLink Universal [FX] / USB MultiLink
□ P&E Cyclone
□ Open Source JTAG

Connect to P&E USB-ML-PPCNEXUS.

Figure 2-51. New Bareboard Project Wizard - Connections Page

7. Check the appropriate connection.

NOTE

You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink QORIVVA** checkbox is checked.

NOTE

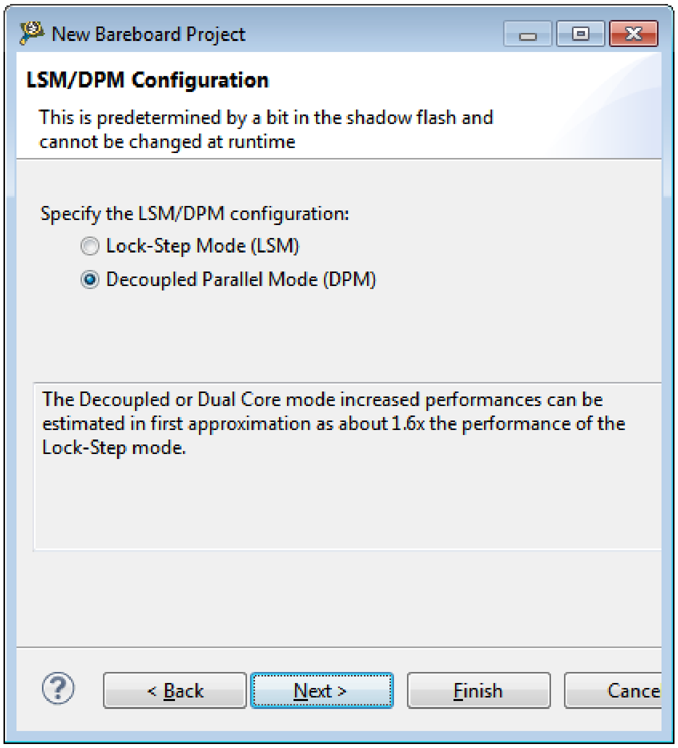
The CodeWarrior debugger supports NEXUS ISTO IEEE 5001-2003 and NEXUS ISTO IEEE 5001-2010 for Qorivva families.

8. Click **Next**.

The **DPM/LSM Configuration** page appears.

NOTE

In case you select *MPC5668E/G Family* derivatives, the **Power Architecture Core Configuration** page appears instead of **DPM/LSM configuration** page.



New Bareboard Project

LSM/DPM Configuration

This is predetermined by a bit in the shadow flash and cannot be changed at runtime

Specify the LSM/DPM configuration:

○ Lock-Step Mode (LSM)

● Decoupled Parallel Mode (DPM)

The Decoupled or Dual Core mode increased performances can be estimated in first approximation as about 1.6x the performance of the Lock-Step mode.

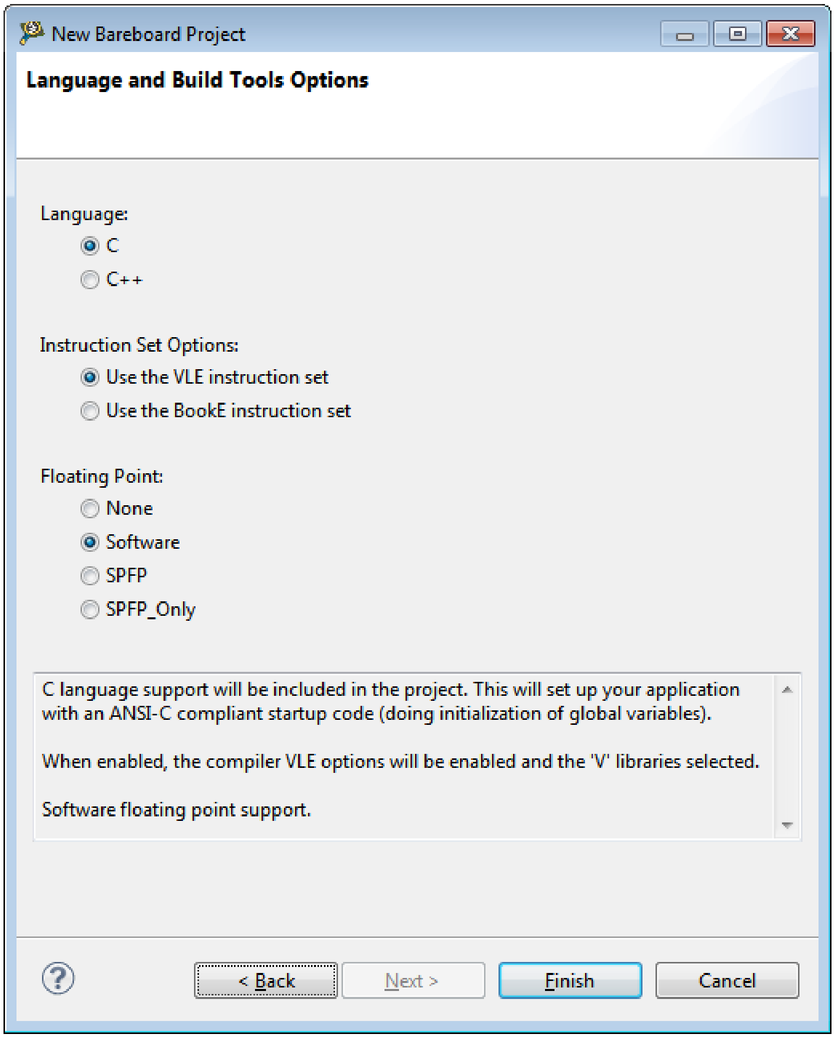
Figure 2-52. New Bareboard Project Wizard - LSM/ DPM Configuration Page

NOTE

This page is available only for *Qorivva*'s MPC56xxK and MPC56xxL derivatives.

9. Click **Next**.

   The **Language and Build Tools Options** appears.



New Bareboard Project

Language and Build Tools Options

Language:

● C

○ C++

Instruction Set Options:

● Use the VLE instruction set

○ Use the BookE instruction set

Floating Point:

○ None

● Software

○ SPFP

○ SPFP_Only

C language support will be included in the project. This will set up your application with an ANSI-C compliant startup code (doing initialization of global variables).

When enabled, the compiler VLE options will be enabled and the 'V' libraries selected.

Software floating point support.

Figure 2-53. New Bareboard Project Wizard - Languages and Build Tools Options Page

10. Select the appropriate options to enable programming language, build tools options, and floating point supports.
11. Click **Finish**.

    The wizard creates a new target board project for the Qorivva architecture. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

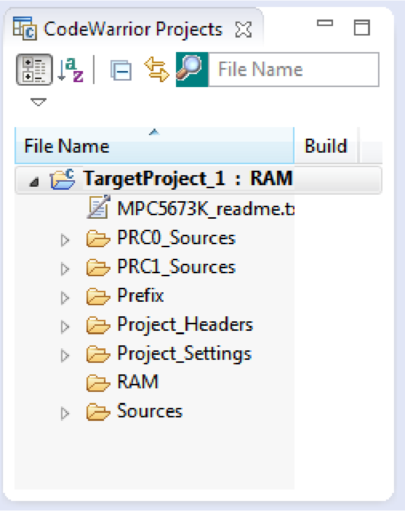


Figure 2-54. CodeWarrior Projects View

NOTE

You can click **Finish** at any step in the Project Wizard to save the project with the default settings.

2.4.2.6 Creating Target Board Projects for 56800/E (DSC)

To create a new target board project for a DSC derivative using the **New Bareboard Project** wizard:

1. Launch the Workbench.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

2. Select **File > New > Bareboard Project**, from the IDE menu bar.

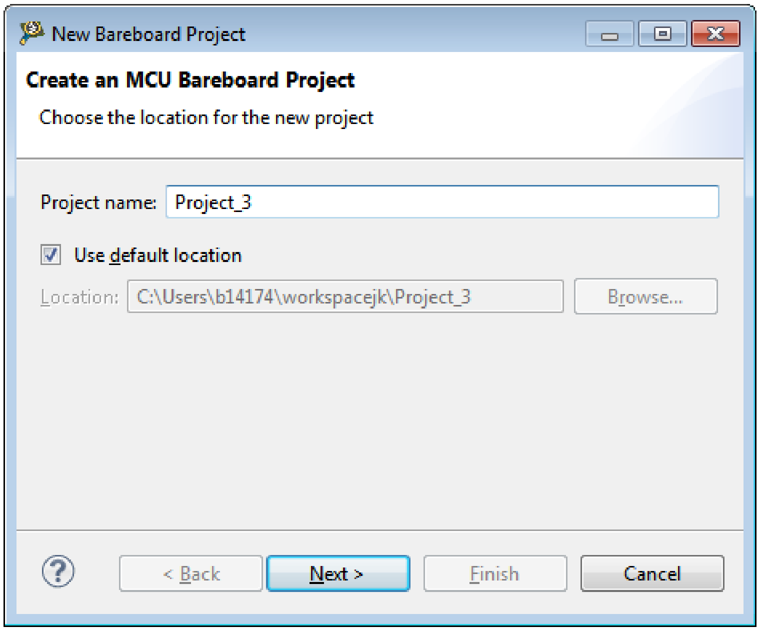
   The **Create an MCU bareboard Project page** of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `Project_3`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the

Browse For Folder dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name: Project_3

☑ Use default location

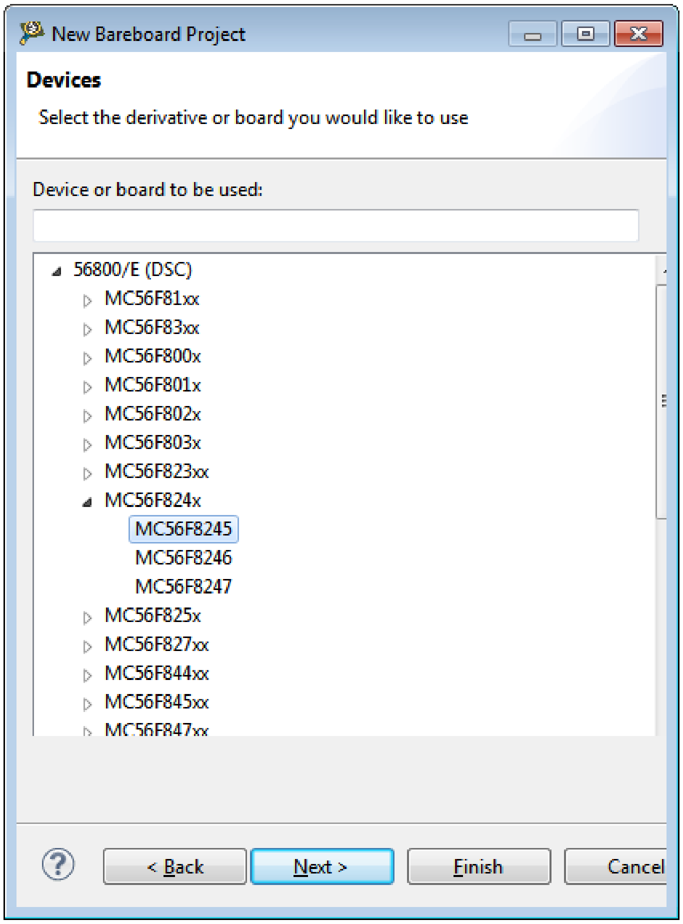
Location: C:\Users\b14174\workspacejk\Project_3

Figure 2-55. New Bareboard Project Wizard - Create an MCU Bareboard Project Page

4. Click **Next**.

   The **Devices** page appears.

5. Expand the desired tree control and select the derivative or board you would like to use. For example, select **56800/E (DSC) > MC56F824x > MC56F8245**.



Devices

Select the derivative or board you would like to use

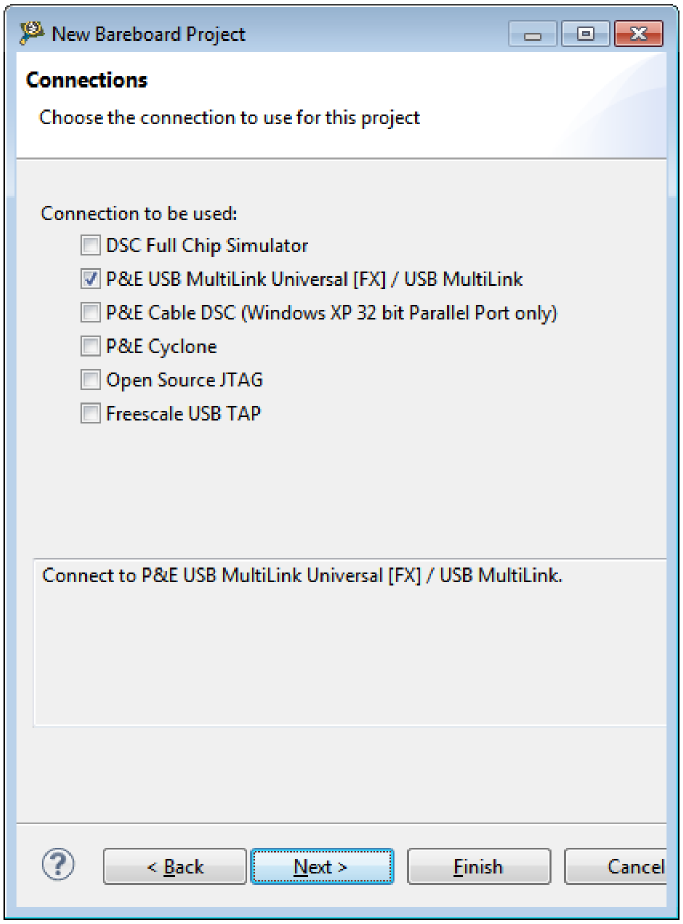
Device or board to be used:

- 56800/E (DSC)
  - MC56F81xx
  - MC56F83xx
  - MC56F800x
  - MC56F801x
  - MC56F802x
  - MC56F803x
  - MC56F823xx
  - MC56F824x
    - MC56F8245
    - MC56F8246
    - MC56F8247
  - MC56F825x
  - MC56F827xx
  - MC56F844xx
  - MC56F845xx
  - MC56F847xx

Figure 2-56. New Bareboard Project Wizard - Devices Page

6. Click **Next**.

   The **Connections** page appears.



Connections

Choose the connection to use for this project

Connection to be used:

- □ DSC Full Chip Simulator
- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cable DSC (Windows XP 32 bit Parallel Port only)
- □ P&E Cyclone
- □ Open Source JTAG
- □ Freescale USB TAP

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-57. New Bareboard Project Wizard - Connections Page

7. Select the desired connection.

NOTE

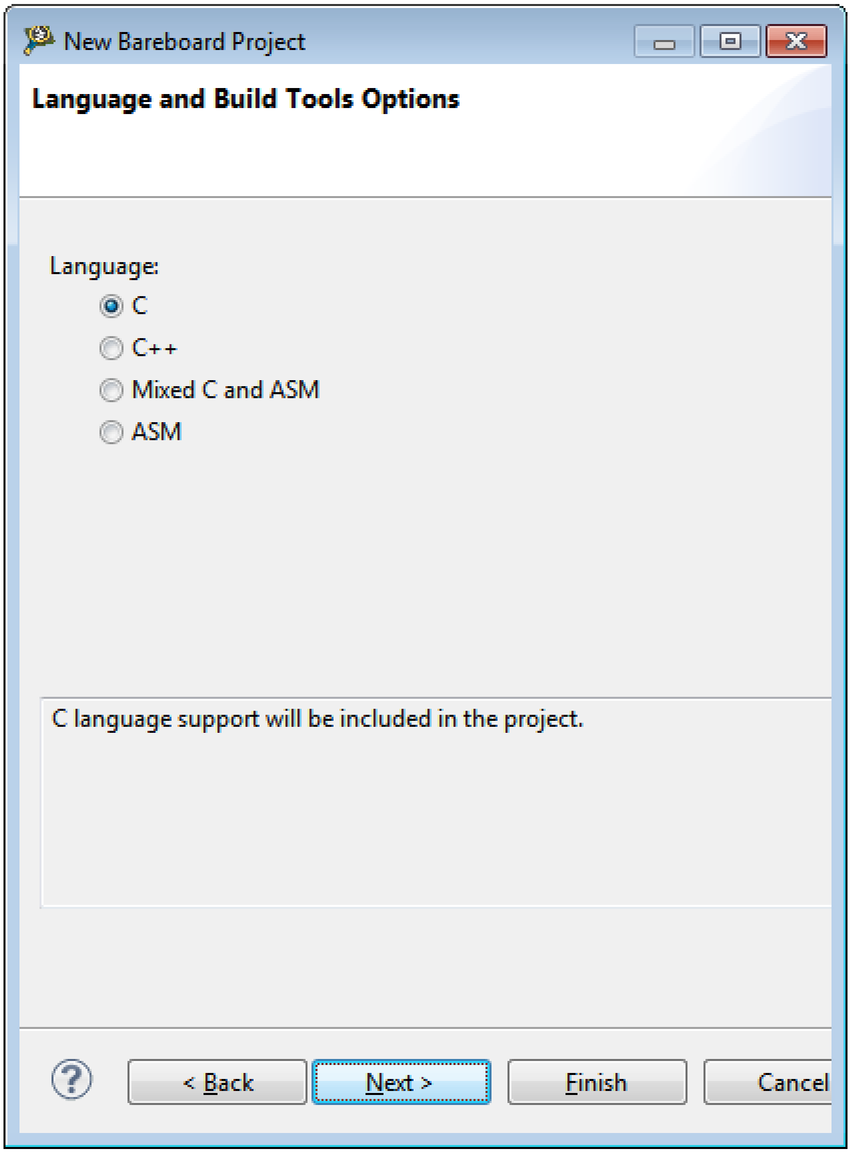
You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink Universal [FX] / USB MultiLink** connection checkbox is checked.

NOTE

The DSC Simulator connection will appear grayed out if any of the MC56F844xx, MC56F827xx, MC56F845xx, or MC56F847xx devices is selected.

8. Click **Next**.

   The **Languages and Build Tools** page appears.



New Bareboard Project

Language and Build Tools Options

Language:

● C

○ C++

○ Mixed C and ASM

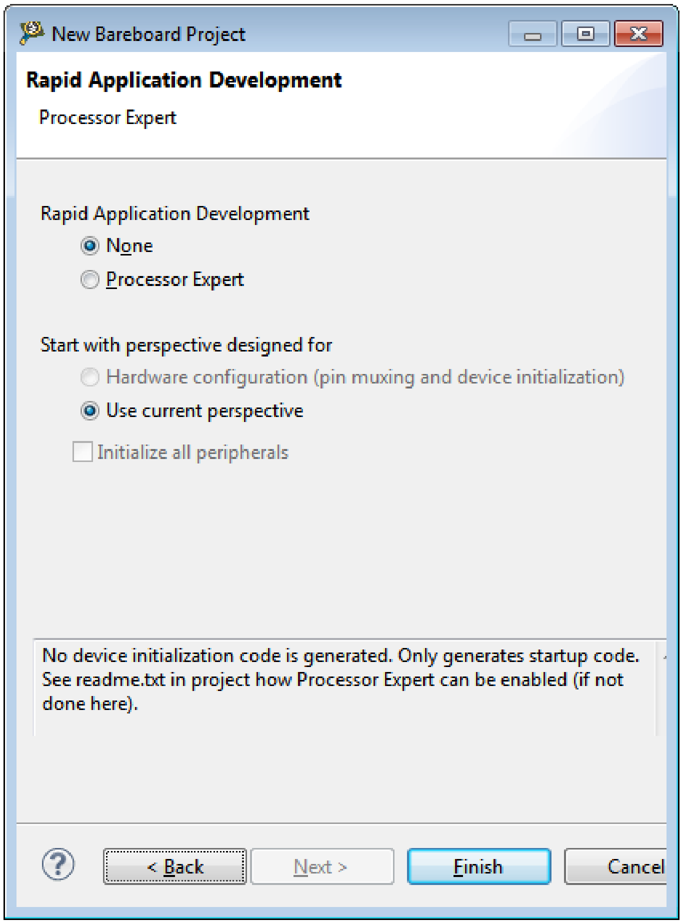
○ ASM

C language support will be included in the project.

Figure 2-58. New Bareboard Project Wizard - Languages Page

9. Select the programming language you want to use. By default, the **Simple C** option is selected.
10. Click **Next**.

    The **Rapid Application Development** page appears.



Rapid Application Development

Processor Expert

Rapid Application Development

● None

○ Processor Expert

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

No device initialization code is generated. Only generates startup code. See readme.txt in project how Processor Expert can be enabled (if not done here).

Figure 2-59. New Bareboard Project Wizard - Rapid Application Development Page

NOTE

The **Processor Expert** option is disabled, if you select the **Simple Assembly** option in the **Connections** page.

11. Select the appropriate option to support rapid application development.
    - **None** - Select to generate only startup code.
    - **Processor Expert** - Select to generate the device initialization code, including low-level drivers.
12. Click **Finish**.

    The wizard creates a new target board project for the DSC architecture according to your specifications. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

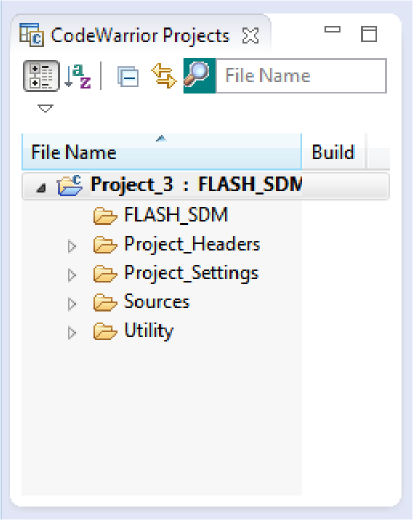


Figure 2-60. CodeWarrior Projects View

The new project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

NOTE

You can click **Finish** at any step in the Project Wizard to save the project with the default settings.

2.4.2.7 Creating Target Board Projects for S12Z

To create a new target board project for an S12Z derivative using the **New Bareboard Project** wizard:

1. Launch the **Workbench**.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

2. Select **File > New > Bareboard Project**, from the IDE menu bar.

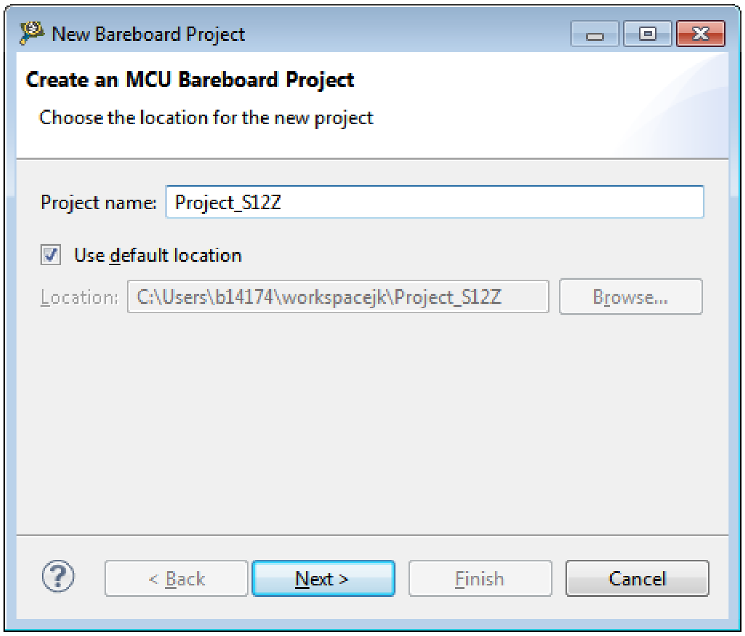
   The **Create an MCU bareboard Project page** of the **New Bareboard Project** wizard appears.

3. Specify a name for the new project. For example, enter the project name as `Project_S12Z`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter

the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.



New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

Project name: Project_S12Z

☑ Use default location

Location: C:\Users\b14174\workspacejk\Project_S12Z

Figure 2-61. New Bareboard Project Wizard - Create an MCU Bareboard Project Page

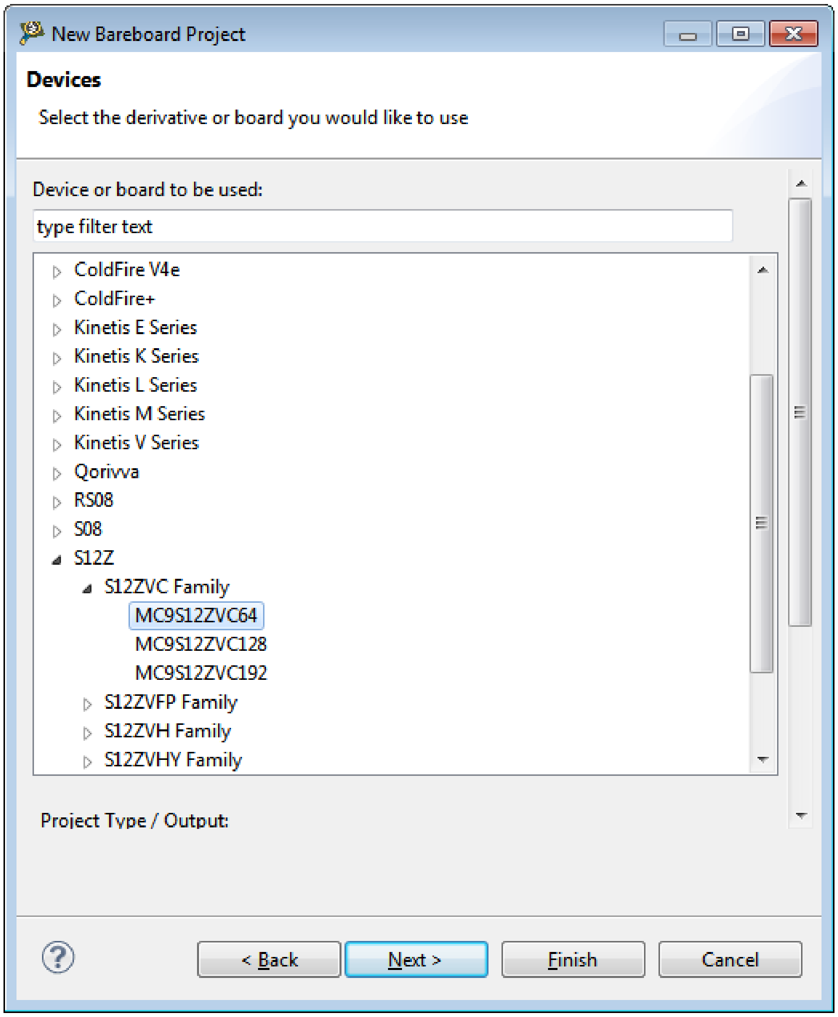
4. Click **Next**.

   The **Devices** page appears.

5. Expand the desired tree control and select the derivative or board you would like to use. For example, select **S12Z > S12ZVM Family > MC9S12ZVMC64**.

NOTE

The product integrates support for MC9S12VM devices (project name Carcassonne). MC9S12ZVML128 - LIN support, MC9S12ZVMC128 - CAN support, MC9S12ZVML64 - LIN support, MC9S12ZVMC64 - CAN support, MC9S12ZVM32 - none.



New Bareboard Project

Devices

Select the derivative or board you would like to use

Device or board to be used:

type filter text

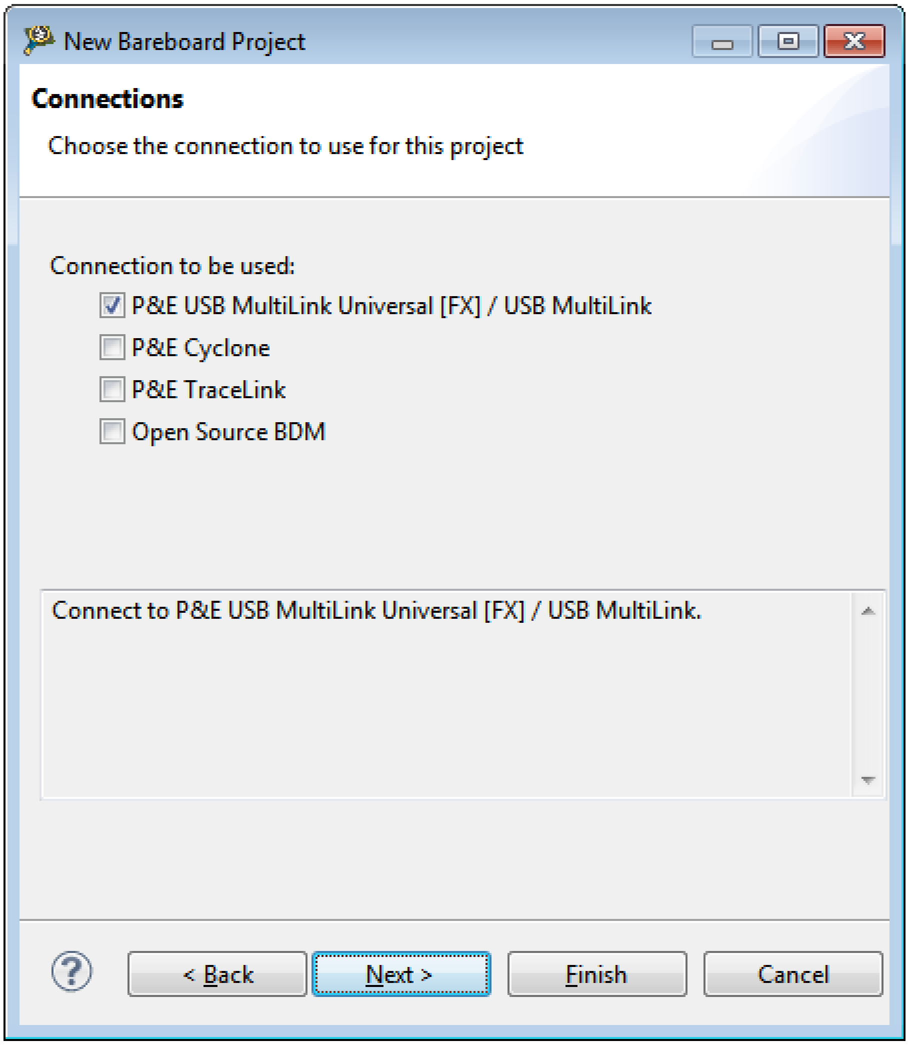
- ColdFire V4e
- ColdFire+
- Kinetis E Series
- Kinetis K Series
- Kinetis L Series
- Kinetis M Series
- Kinetis V Series
- Qorivva
- RS08
- S08
- S12Z
  - S12ZVC Family
    - MC9S12ZVC64
    - MC9S12ZVC128
    - MC9S12ZVC192
  - S12ZVFP Family
  - S12ZVH Family
  - S12ZVHY Family

Project Type / Output:

Figure 2-62. New Bareboard Project Wizard - Devices Page

6. Click **Next**.

   The **Connections** page appears.



New Bareboard Project

Connections

Choose the connection to use for this project

Connection to be used:

☑ P&E USB MultiLink Universal [FX] / USB MultiLink
☐ P&E Cyclone
☐ P&E TraceLink
☐ Open Source BDM

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 2-63. New Bareboard Project Wizard - Connections Page

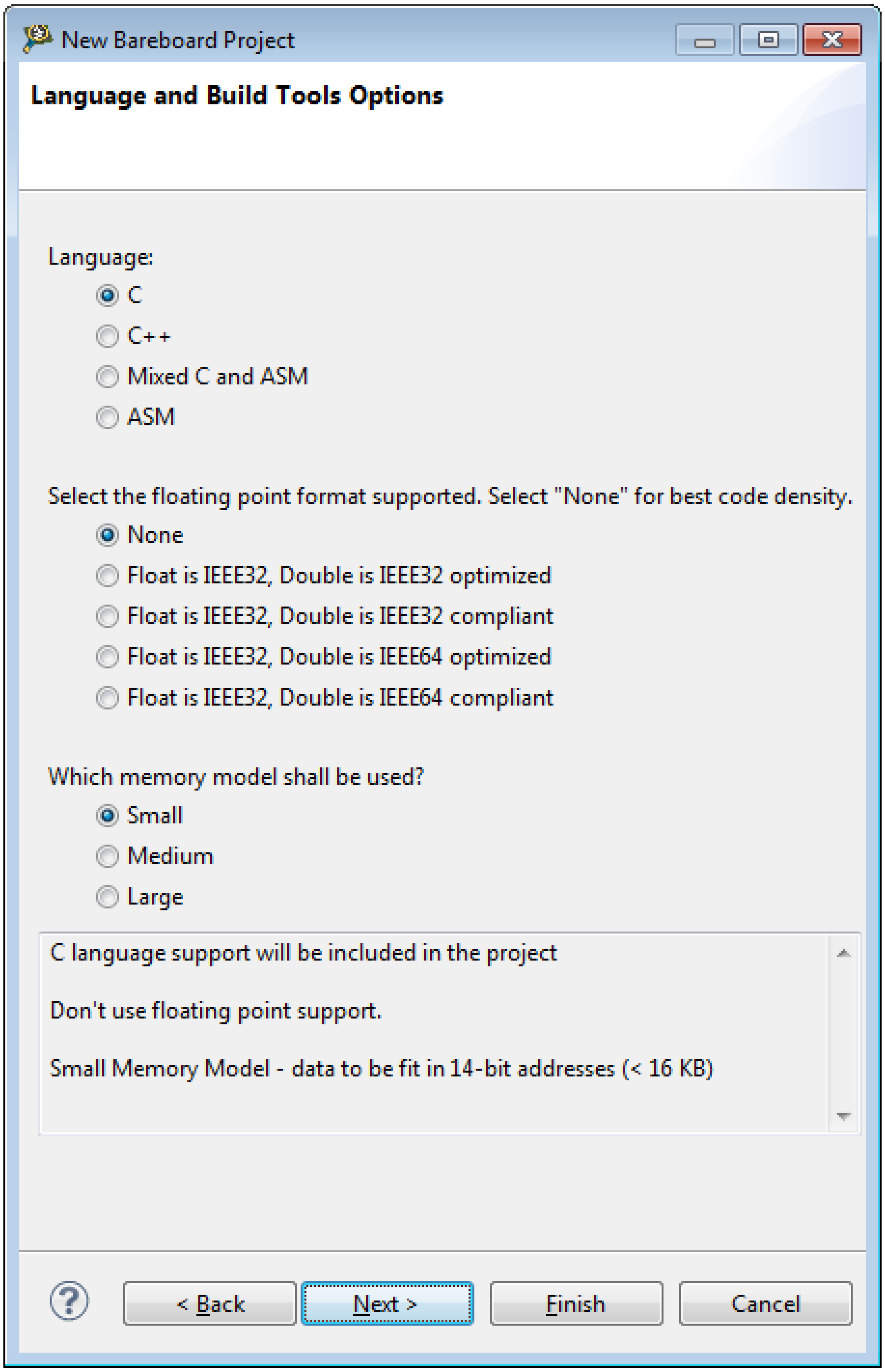
7. Select the desired connection.

NOTE

You can select multiple connections by checking appropriate checkboxes in the **Connections** page. By default, the **P&E USB MultiLink Universal [FX] / USB MultiLink** connection checkbox is checked.

8. Click **Next**.

   The **Languages and Build Tools Options** page appears.



Language and Build Tools Options

Language:

● C

○ C++

○ Mixed C and ASM

○ ASM

Select the floating point format supported. Select "None" for best code density.

● None

○ Float is IEEE32, Double is IEEE32 optimized

○ Float is IEEE32, Double is IEEE32 compliant

○ Float is IEEE32, Double is IEEE64 optimized

○ Float is IEEE32, Double is IEEE64 compliant

Which memory model shall be used?

● Small

○ Medium

○ Large

C language support will be included in the project

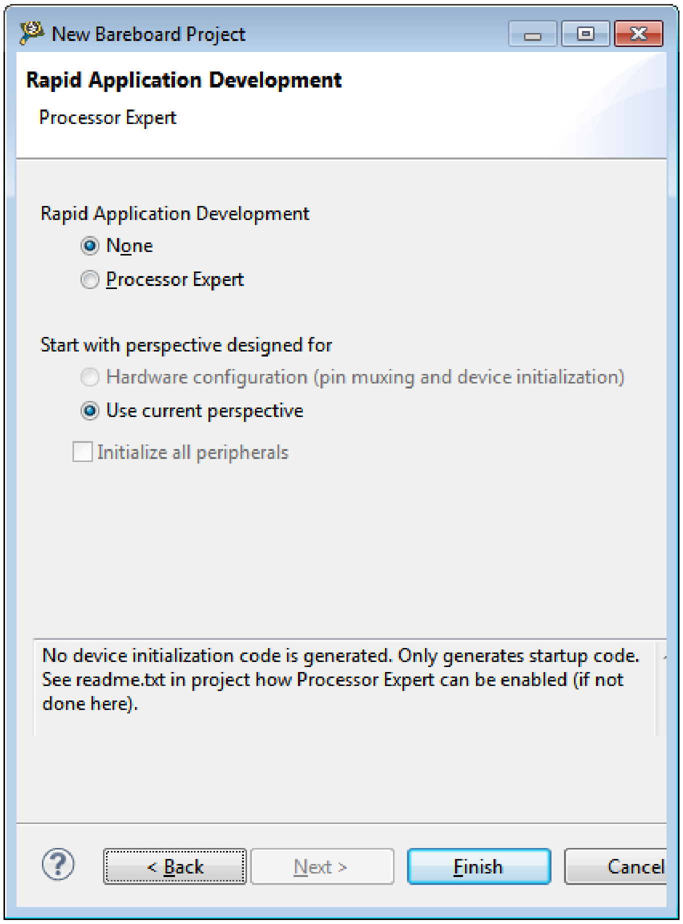
Don't use floating point support.

Small Memory Model - data to be fit in 14-bit addresses (< 16 KB)

Figure 2-64. New Bareboard Project Wizard - Languages and Build Tools page

9. Select the programming language you want to use. By default, the **Simple C** option is selected.
10. Select the floating point format you want to use. Selecting None provides the best code density.
11. Select the memory model to use.
12. Click **Next.**

The Rapid Application Development page appears.



New Bareboard Project

Rapid Application Development

Processor Expert

Rapid Application Development

● None

○ Processor Expert

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

No device initialization code is generated. Only generates startup code. See readme.txt in project how Processor Expert can be enabled (if not done here).

Figure 2-65. New Bareboard Project Wizard - Rapid Application Development Page

NOTE

The **Processor Expert** option is disabled, if you select the C++, or ASM language option in the Language and Build Tools Options page.

13. Select the appropriate option to support rapid application development.
    - **None** - Select to generate only startup code.
    - **Processor Expert** - Select to generate the device initialization code, including low-level drivers.
14. Click **Finish**.

    The wizard creates a new target board project for the S12Z architecture according to your specifications. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

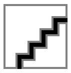


Figure 2-66. CodeWarrior Projects View

The new project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

NOTE

You can click **Finish** at any step in the Project Wizard to save the project with the default settings.

2.4.3 Creating Linux/uClinux Application Project

To create a new Linux/uClinux Application Project using the **New Linux/uClinux Application Project** wizard:

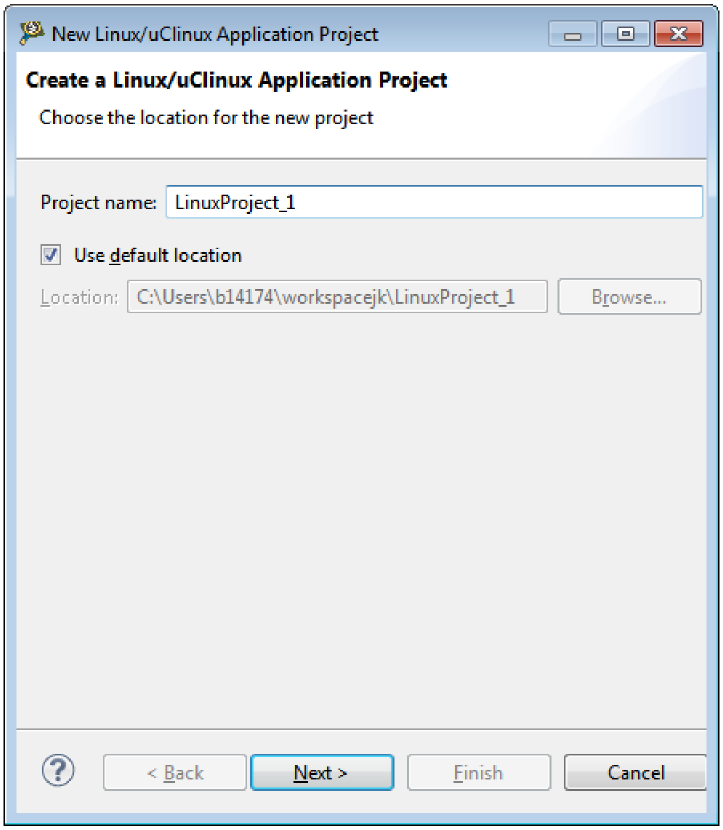
1. Launch the **Workbench**.

   NOTE

   For information about launching the Workbench, refer to the topic Launching Workbench.

2. Select **File > New > Linux/uClinux Application Project**, from the IDE menu bar.

   The Create a Linux/uClinux Application Project page of the **New Linux/uClinux Application Project** wizard appears.



New Linux/uClinux Application Project

Create a Linux/uClinux Application Project

Choose the location for the new project

Project name: LinuxProject_1

☑ Use default location

Location: C:\Users\b14174\workspacejk\LinuxProject_1

Figure 2-67. Create a Linux/uClinux Application Project Page

3. Specify a name for the new project. For example, enter the project name as `LinuxProject_1`.

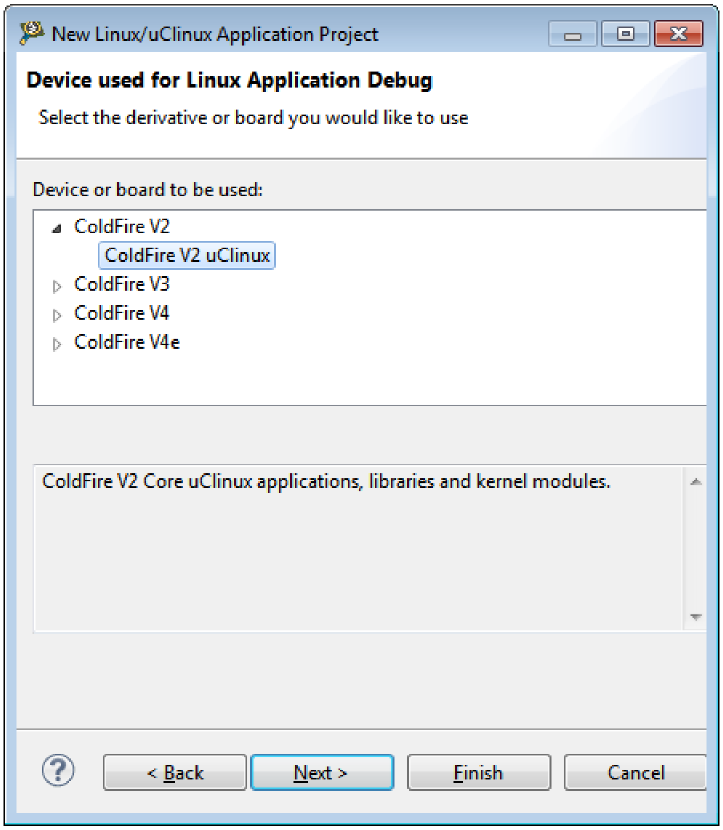
NOTE

> If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.

4. Click **Next**.

   The **Device used for Linux Application Debug** page appears.

5. Expand the tree control and select the derivative or board you would like to use. For example, select **ColdFire V2 > ColdFire V2 uClinux**.



New Linux/uClinux Application Project

Device used for Linux Application Debug

Select the derivative or board you would like to use

Device or board to be used:

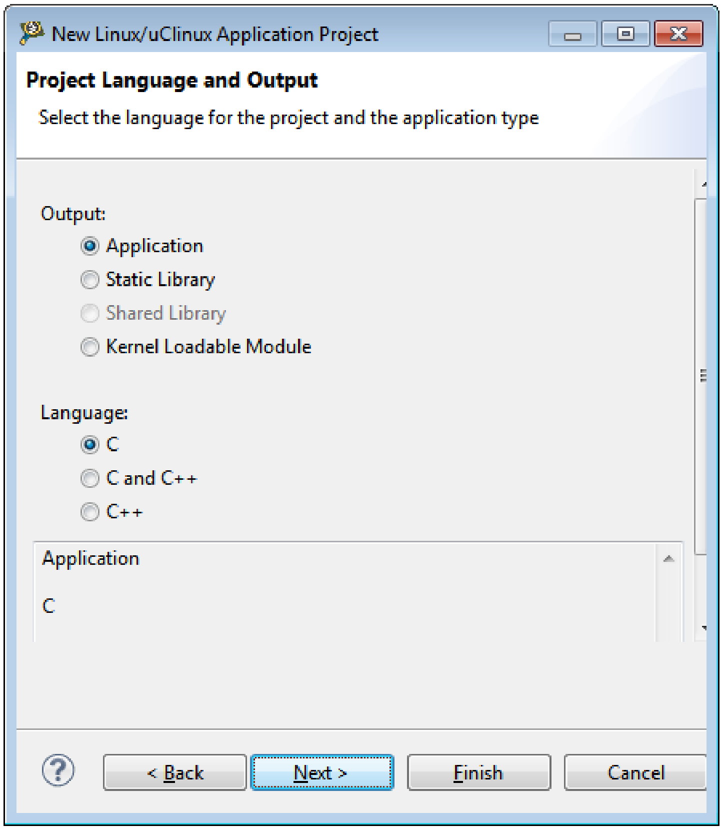
- ColdFire V2
  - ColdFire V2 uClinux
- ColdFire V3
- ColdFire V4
- ColdFire V4e

ColdFire V2 Core uClinux applications, libraries and kernel modules.

Figure 2-68. Device used for Linux Application Debug Page

6. Click **Next**.

   The **Project Language and Output** page appears.



New Linux/uClinux Application Project

Project Language and Output

Select the language for the project and the application type

Output:

- ● Application
- ○ Static Library
- ○ Shared Library
- ○ Kernel Loadable Module

Language:

- ● C
- ○ C and C++
- ○ C++

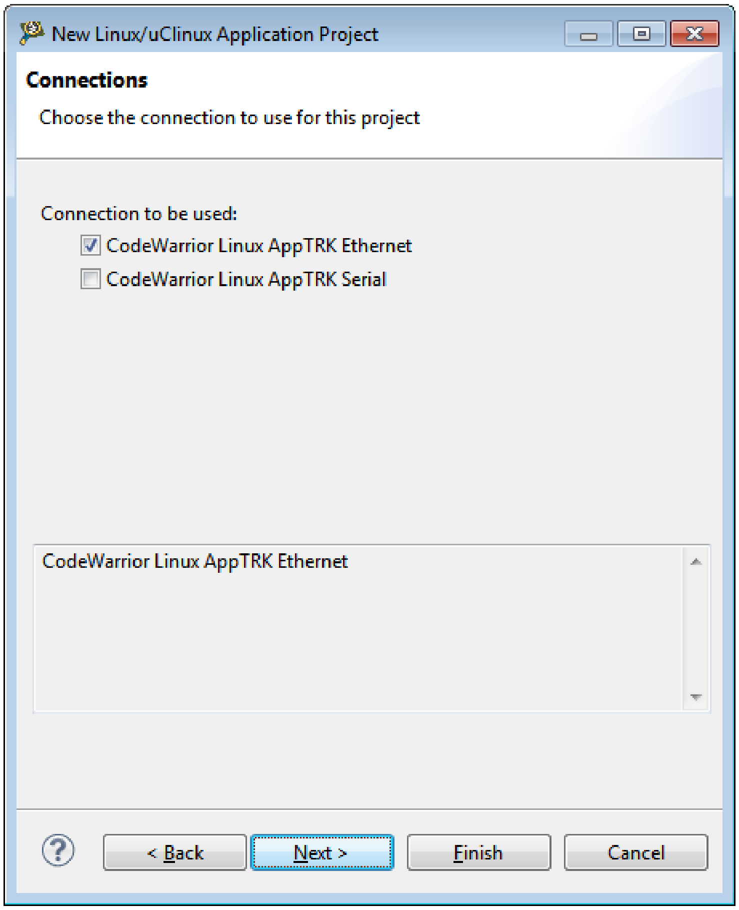
Application

C

Figure 2-69. Project Language and Output Page

7. Select the output type and the programming language you want to use for this project. For example, select **Application** and **C and C++**.
8. Click **Next**.

   The **Connections** page appears.



New Linux/uClinux Application Project

Connections

Choose the connection to use for this project

Connection to be used:

☑ CodeWarrior Linux AppTRK Ethernet

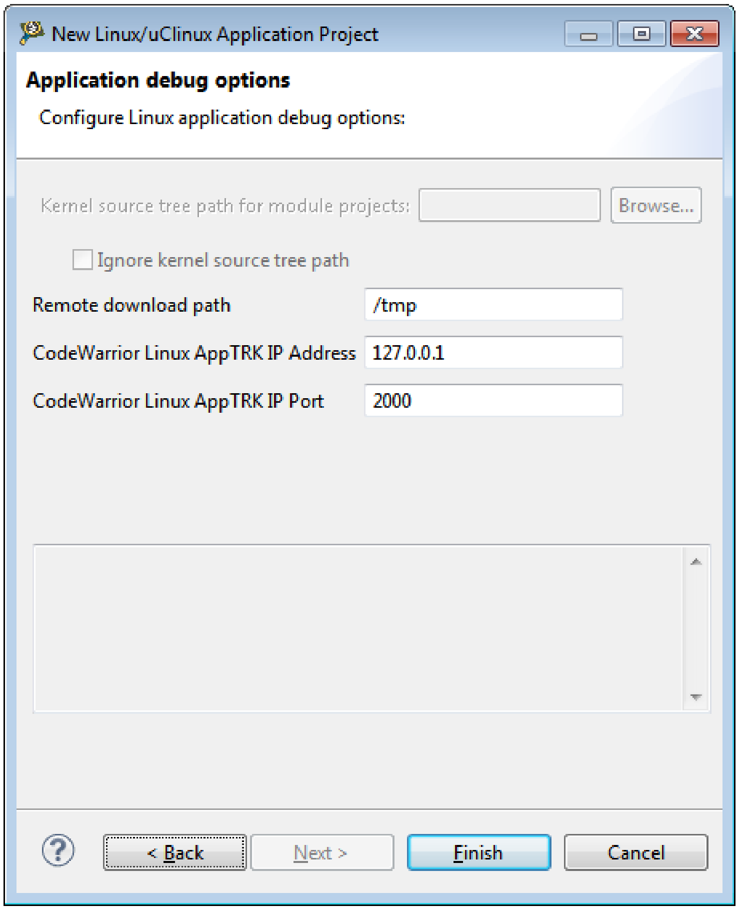
□ CodeWarrior Linux AppTRK Serial

CodeWarrior Linux AppTRK Ethernet

Figure 2-70. Connections Page

9. Check the appropriate connection.
10. Click **Next**.

    The **Application debug options** page appears.



New Linux/uClinux Application Project

Application debug options

Configure Linux application debug options:

Kernel source tree path for modules projects: Browse...

□ Ignore kernel source tree path

Remote download path: /tmp

CodeWarrior Linux AppTRK IP Address: 127.0.0.1

CodeWarrior Linux AppTRK IP Port: 2000

Figure 2-71. Application Debug Options Page

11. From the list, select the method with which you want the IDE to connect to the target system.
12. In the **Remote download path** text box, specify the path. By default, it is `/tmp`.
13. In the **CodeWarrior Linux AppTRK IP Address** and **CodeWarrior Linux AppTRK IP Port** text boxes, enter the IP address and listening port of the target system. By default, the value of the AppTRK IP address is `127.0.0.1` and the AppTRK IP port number is `2000`.
14. Click **Finish**.

    The wizard closes. The IDE generates a new project according to your specifications. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

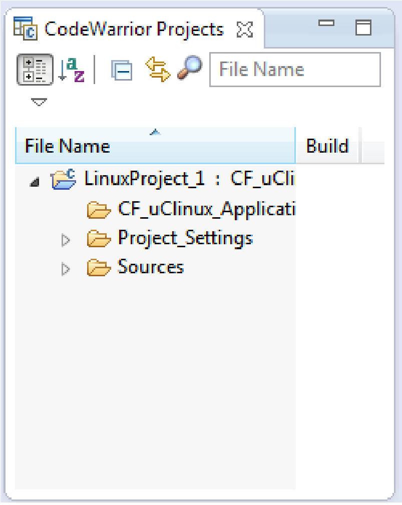


Figure 2-72. CodeWarrior Projects View

2.5 Building Projects

In large workspaces, building the entire workspace can take a long time if you make changes with a significant impact on dependent projects. Often there are only a few projects that really matter to you at a given time.

To build only the selected projects, and any prerequisite projects that need to be built in order to correctly build the selected projects, select **Project > Build Project** from the CodeWarrior IDE menu bar.

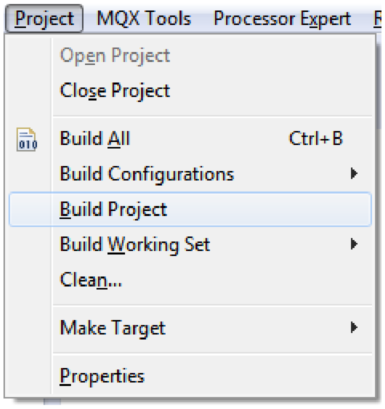


Figure 2-73. Project Menu - Build Project

Alternatively, select **Project > Build All**. .

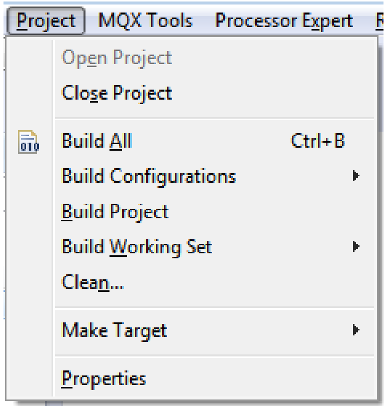


Figure 2-74. Project Menu - Build All

Alternatively, you can use the options in the **Commander** view.

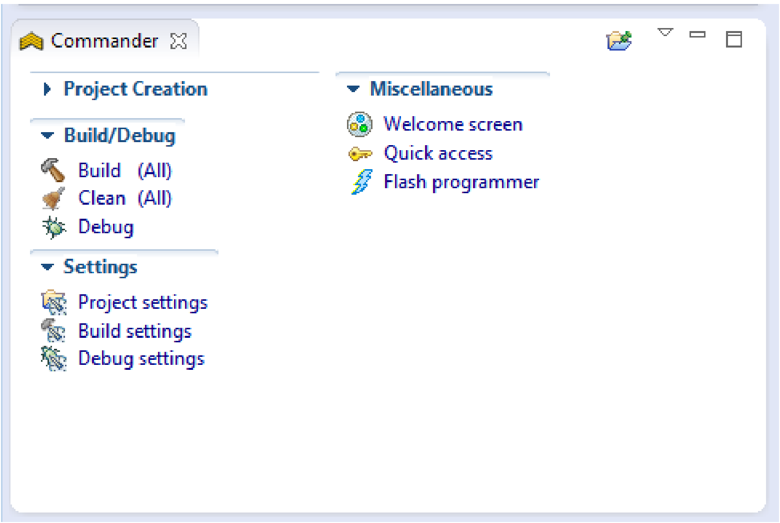


Figure 2-75. Commander View

2.6 Debugging Projects

When you use the **New Bareboard Project** wizard to create a new project, the wizard sets the debugger settings of the project's launch configurations to default values. You can change these default values based on your requirements.

To debug a project, perform these steps.

1. Launch the IDE.
2. From the main menu bar of the IDE, select **Run > Debug Configurations** . The IDE uses the settings in the launch configuration to generate debugging information and initiate communications with the target board.

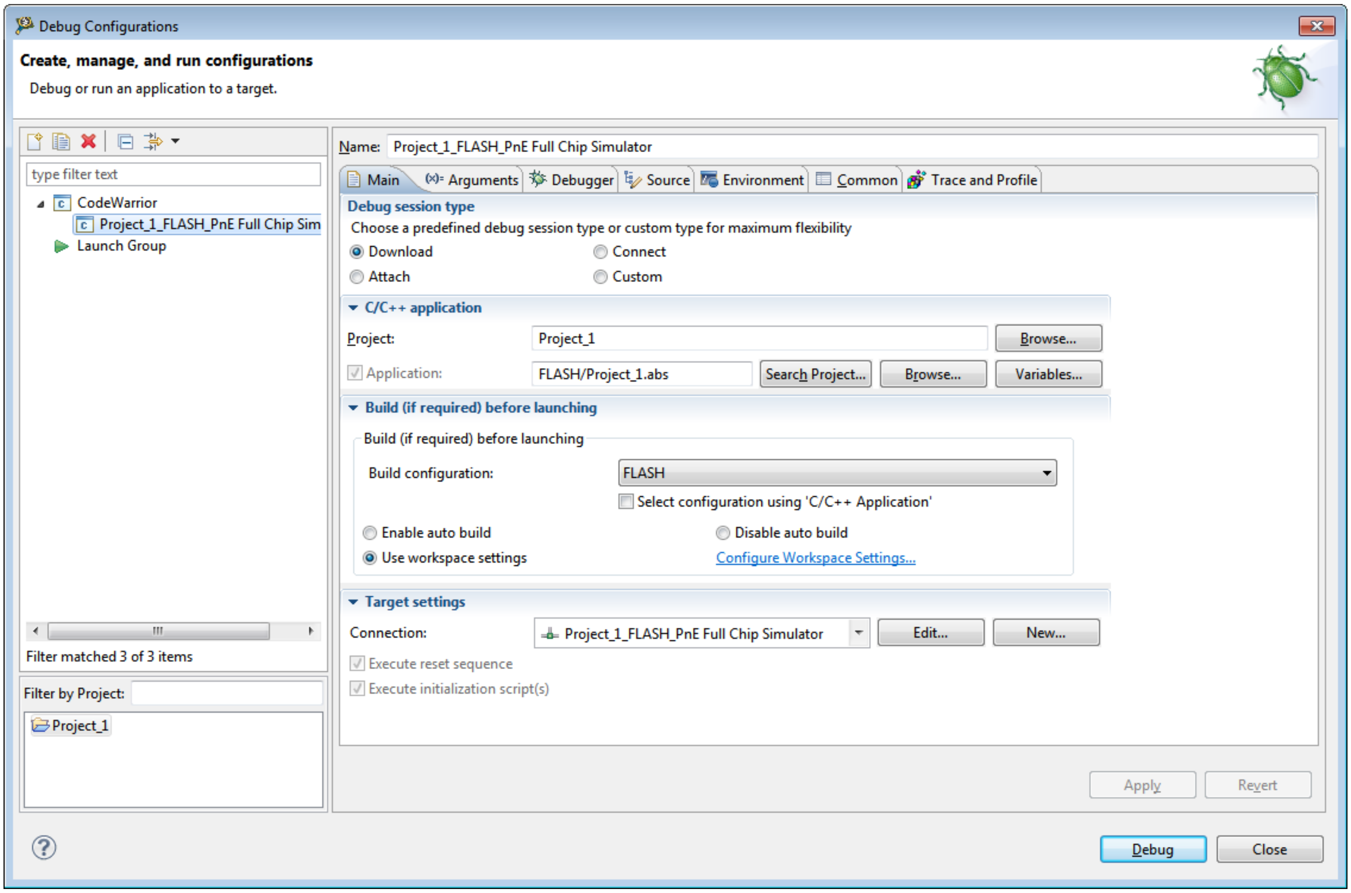
The **DebugConfigurations** dialog box appears. The left side of this dialog box has a list of debug configurations that apply to the current application.

NOTE

For more information on how to use the debugger, refer to the CodeWarrior Common Features Guide and the Working with Debugger chapter of this manual.

3. Expand the **CodeWarrior Download** configuration.
4. From the expanded list, select the debug configuration that you want to modify.

The figure below displays the **Debug Configurations** dialog box with the settings for the debug configuration you selected.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - Project_1_FLASH_PnE Full Chip Sim
- Launch Group

Filter matched 3 of 3 items

Filter by Project:

Project_1

Name: Project_1_FLASH_PnE Full Chip Simulator

Main | Arguments | Debugger | Source | Environment | Common | Trace and Profile

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: Project_1 [Browse...]

☑ Application: FLASH/Project_1.abs [Search Project...] [Browse...] [Variables...]

Build (if required) before launching

Build (if required) before launching

Build configuration: FLASH

☐ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Target settings

Connection: Project_1_FLASH_PnE Full Chip Simulator [Edit...] [New...]

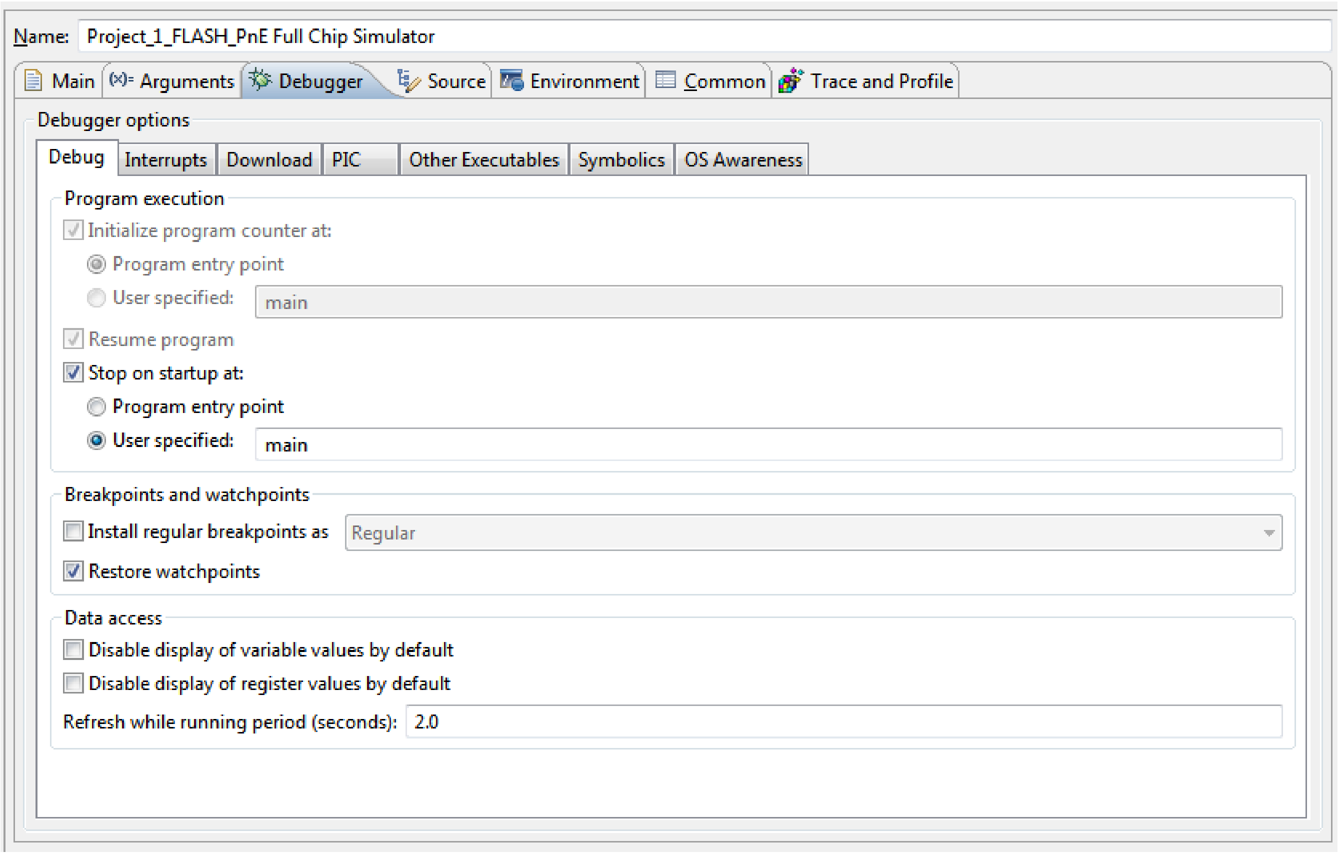
☑ Execute reset sequence

☑ Execute initialization script(s)

Figure 2-76. Debug Configurations Dialog Box

5. Click the **Debugger** tab.

The **Debugger** page appears in the area beneath the tabs.



Name: Project_1_FLASH_PnE Full Chip Simulator

Debugger options

Program execution

☑ Initialize program counter at:

● Program entry point

○ User specified: main

☑ Resume program

☑ Stop on startup at:

○ Program entry point

● User specified: main

Breakpoints and watchpoints

□ Install regular breakpoints as Regular

☑ Restore watchpoints

Data access

□ Disable display of variable values by default

□ Disable display of register values by default

Refresh while running period (seconds): 2.0

Figure 2-77. Debug Configurations Dialog Box - Debugger Page

6. Change the settings on this page as per your requirements.

NOTE

For more information on debugger, refer to the chapter Working with Debugger.

7. Click **Apply** to save the new settings.
8. Click **Debug** to start the debugging session.

You just finished starting a debugging session and attaching the debugger to a process.

NOTE

You can click **Revert** to undo any of the unsaved changes. The IDE restores the last set of saved settings to all pages of the **Debug Configurations** dialog box. Also, the IDE disables **Revert** until you make new pending changes.

2.7 Deleting Projects

To delete a project, follow these steps.

1. Select the project you want to delete in the **CodeWarrior Projects** view.
2. Select **Edit** > **Delete**.

   The **Delete Resources** dialog box appears.

   NOTE
   Alternatively, you can also select **Delete** from the context menu when you right-click on the project.

3. Check the **Delete project contents on disk (cannot be undone**) checkbox if you want to delete the contents of the selected project. Else, clear the **Delete project contents on disk (cannot be undone**) checkbox.

   NOTE
   You will not be able to restore your project using *Undo*, if you select the **Delete project contents on disk (cannot be undone)** option.

4. Click **OK**.

   The project is removed from the **CodeWarrior Projects** view.

2.8 Porting Classic DSC Project to Eclipse Project

The CodeWarrior Classic Project Importer feature in Eclipse helps automate the conversion of a legacy DSC project to an Eclipse CDT project.

This feature lets you:

- select the classic CodeWarrior project,
- set targets to import,
- configure source trees and shielded folders,
- edit access paths for each target,
- list files that are not found in the previous settings,
- specify the new Eclipse project name and location,
- list warning or errors in the conversion process, and
- open the newly created Eclipse project.

NOTE

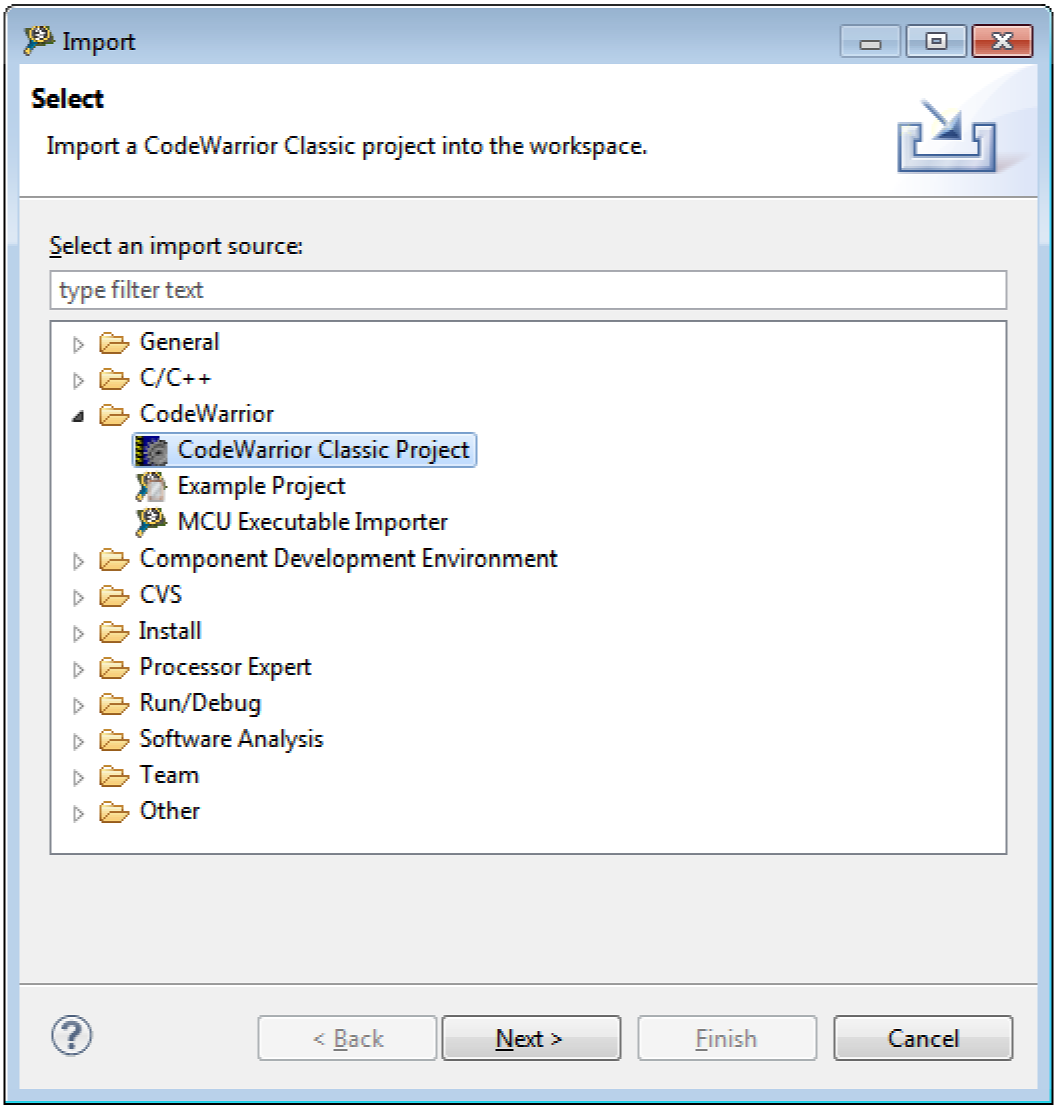
Before starting the process ensure that the CodeWarrior DSC project you want to import has all of its files, such as the source, linker command, and settings file.

To port a classic DSC project, perform these steps.

1. Select **File > Import** from the **Workbench** menu bar.

   The **Import** dialog box appears.

2. Expand the **CodeWarrior** tree control and select **CodeWarrior Classic Project**.



Select

Import a CodeWarrior Classic project into the workspace.

Select an import source:

type filter text

Figure 2-78. Import Dialog Box

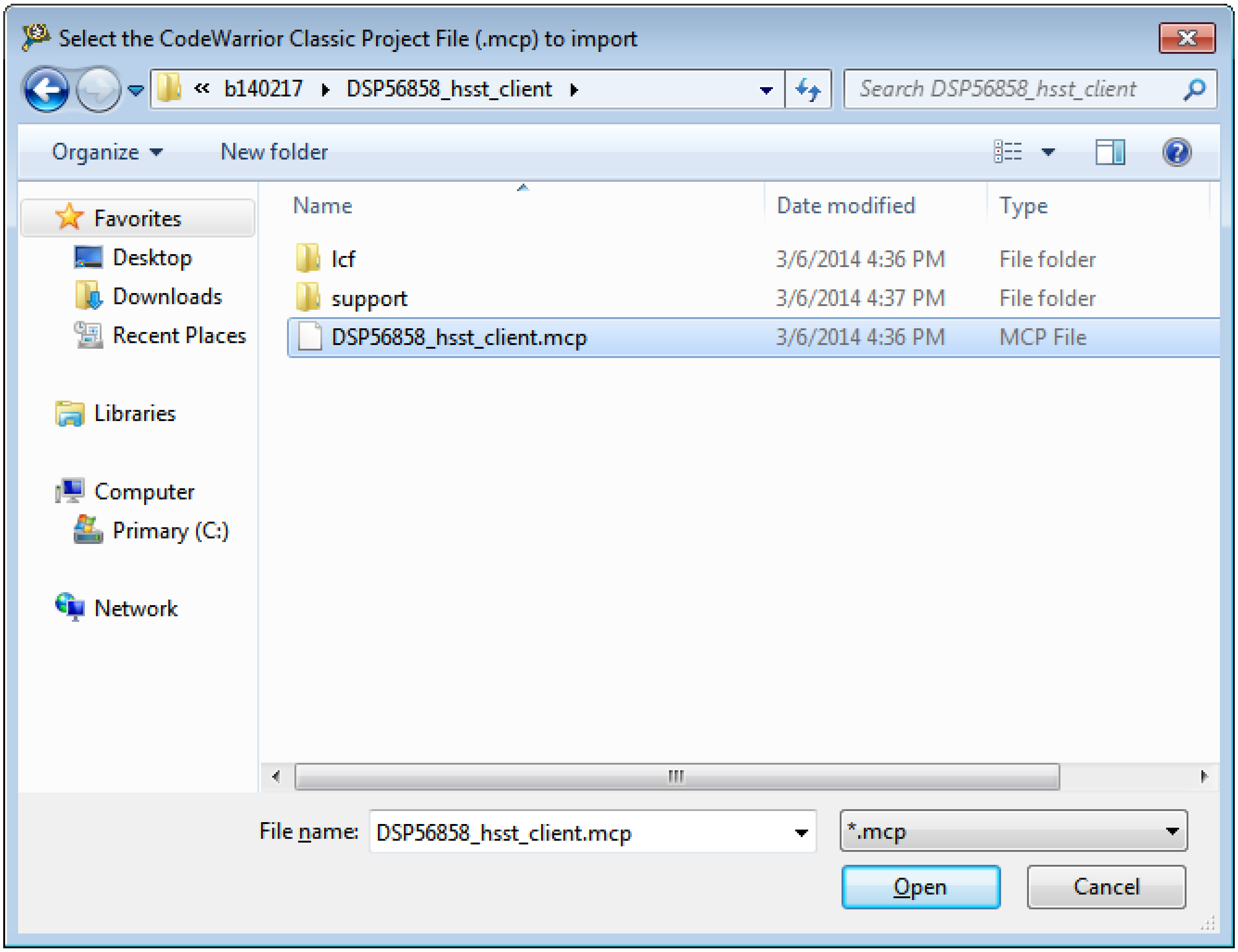
3. Click **Next**.

   The first page of the **CodeWarrior Classic Project** importer wizard appears.

4. Enter the path and name of the classic CodeWarrior project file to import in the **Project file** text box. Alternatively, click **Browse** and use the **Select The CodeWarrior Project File to Import** dialog box to select the project file to import.

Tip

The project file has an extension of `.mcp`. Select the `.mcp` file.


Select the CodeWarrior Classic Project File (.mcp) to import
« b140217 ▸ DSP56858_hsst_client ▸
Search DSP56858_hsst_client
Organize ▾
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Computer
Primary (C:)
Network
Open
Cancel

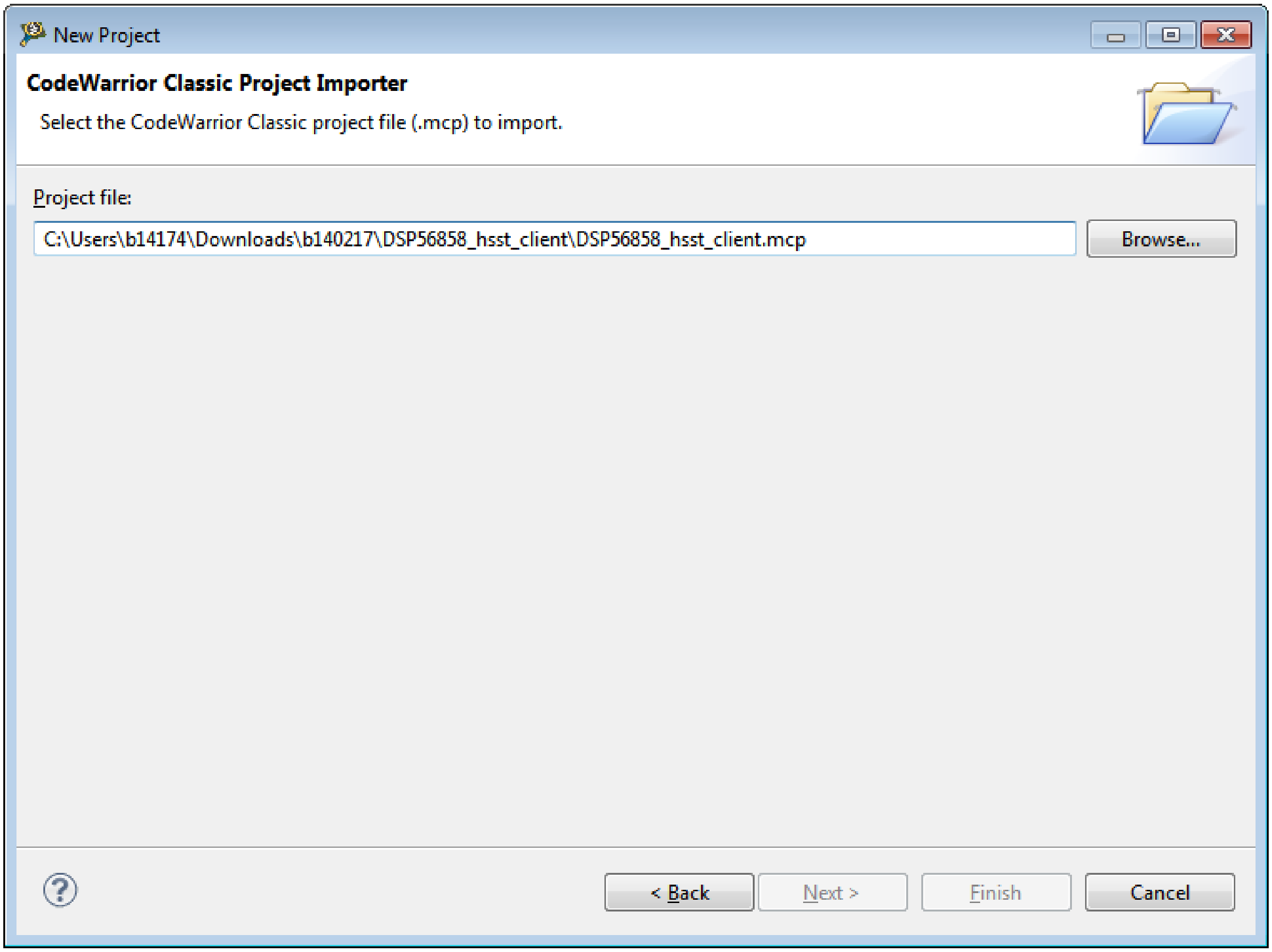
Name	Date modified	Type
lcf	3/6/2014 4:36 PM	File folder
support	3/6/2014 4:37 PM	File folder
DSP56858_hsst_client.mcp	3/6/2014 4:36 PM	MCP File

File name: DSP56858_hsst_client.mcp

*.mcp

Figure 2-79. Select The CodeWarrior Project File to Import Dialog Box

The path of the project file to import appears in the **Project file** text box.



CodeWarrior Classic Project Importer

Select the CodeWarrior Classic project file (.mcp) to import.

Project file:

C:\Users\b14174\Downloads\b140217\DSP56858_hsst_client\DSP56858_hsst_client.mcp

Figure 2-80. Path of CodeWarrior Project File to Import

5. Click **Next**.

   The **Options** page of the **CodeWarrior Classic Project Importer** wizard appears.

6. Select the build target that uses the DSC toolchain you want the generated Eclipse project to use, from the **Toolchain Target** list box.

   ### NOTE

   The toolchain target linker in the classic project defines the project type of the generated Eclipse project, including the toolchain and build settings.

   The build targets table displays all the targets discovered in the project file and is used to generate equivalent Eclipse build configurations.

7. You can import each build target in the classic CodeWarrior project based upon predefined configurations of the toolchain.
8. If you want to import sub-projects included in the classic CodeWarrior project, check the **Recursively Import Sub-Projects** checkbox.

The **CodeWarrior Classic Project** wizard imports the sub-projects with the main project.

NOTE

The **CodeWarrior Classic Project** wizard will copy only those files that are displayed in the **CodeWarrior Projects view**. The wizard will not import a file if it is not displayed or does not include project information.

9. Click **Advanced**.

   The **Advanced Options** dialog box appears.

   WARNING

   Checking the **Copy files into the new project** checkbox may cause build errors.

10. Check the **Copy files into the new project** checkbox.
11. Click **OK** to close the dialog box.
12. Click **Next**.

    The **Globals** page of the **CodeWarrior Classic Project** wizard appears. This page lets you edit global settings that can effect how the project's build options are imported.

    The table below lists the options on the **CodeWarrior Classic Project Importer - Globals** page.

Table 2-24. CodeWarrior Project Importer - Globals Page Options

Options	Description
Shielded Folder List	Lists the concealed contents of folders from the IDE's search operations during a build. This was done by placing special characters in the directory name. For example, sample code was concealed in a (CodeWarrior Examples) folder. The **Shielded Folder List** table lists these options: • \(.*\) • CVS • .*[_]Data You use the **Add, Delete**, and **Clear** buttons to modify the information in this list. The table below lists these buttons with their descriptions.
Sources	Specifies the location of the source trees. If an access path is defined relative to a source tree, the source tree should be listed in this table. The **{Project}** source tree is defined automatically. The **Sources Trees** table lists these options:

Table 2-24. CodeWarrior Project Importer - Globals Page Options

Options	Description
	• Name - Lists the source name. For example: Compiler. • Compiler - Lists the path source name. For example: `<CW Install>/MCU.` You use the **Add**, **Delete**, and **Clear** buttons to modify the information in this list. The table below lists these buttons with their descriptions.

Table 2-25. CodeWarrior Project Importer - Globals Page Buttons

Button	Description
Add	Add a new entry to the list.
Delete	Deletes the selected item.
Clear	Clears the entire list.

13. To add a new expression to the **Shielded Folder List** table, perform these steps.
    a. Click **Add**.

    (regular_expression) appears in the shielded folder list.

    b. Double-click *(regular_expression)* and type the required expression.

    The new expression appears in shielded folder list.

14. To delete an existing expression from the **Shielded Folder List** table, select the expression and click **Delete**.

    The selected expression is deleted from the shielded folder list.

15. To remove all the existing expressions from the **Shielded Folder List** table, click **Clear**.

    All the expressions are deleted from the shielded folder list.

16. To add a new source to the **Source Trees** table, perform these steps.
    a. Click **Add**.

    SourceName appears in the source trees list.

    b. Double-click *SourceName.*

    The **Edit Table Values** dialog box appears.

    c. In the **Name** text box, enter the source name.

d. In the **Path** text box, enter the path of the new source. Alternatively, click **Browse** and use the **Browse For Folder** dialog box and navigate to the required source.

e. Click **OK**.

The new source appears in **Source Trees** list.

17. Click **Next**.

The **CodeWarrior Project Importer - Access Paths** page appears.

NOTE

Access paths are directory paths the CodeWarrior tools use to search for libraries, runtime support files, and other object files.

The table below lists the options on the **CodeWarrior Project Importer - Access Paths** page.

Table 2-26. CodeWarrior Project Importer - Access Paths Page

Option	Description
Target	Lets you select the build target whose access paths you want to modify.
Access Path Table	Lists the access paths used by the build target selected in the **Build Target** list box. Each row lists: • **Path** - Directory path. • **Recursive** - Whether the path is searched recursively. For example: `false` or `true`. • **Type** - Specify path to be searched. For example: `user` or `system`. • **Error** - Unresolved access paths, marked as "X", if any. You use the **Add**, **Delete**, and **Clear** buttons to modify the information in this list. The table below lists these buttons with their descriptions.

Table 2-27. CodeWarrior Project Importer - Access Paths Page Buttons

Button	Description
Add	Add a new directory path to the list.
Delete	Deletes the selected directory path from the list.
Clear	Clears the entire list.

18. To add a new directory path to list, perform these steps.

a. Click **Add**.

(C:\SourcePath) appears in the **Path** list.

b. Double-click *(C:\SourcePath)*.

The **Edit Table Values** dialog box appears.

c. In the **Path** text box, enter the directory path. Alternatively, click **Browse** and use the **Browse For Folder** dialog box and navigate to the required source.
d. From the **Recursive** list box, select `false` or t`rue`.
e. From the **Type** list box, select `user` or `system`.
f. Click **OK**.

The new access path appears in table.

19. Click **Next**.

The **CodeWarrior Project Importer - Problems** page appears. This page displays the project files that the wizard could not locate. You can use the Build Target list box to select another build target and view the missing files.

20. To locate the missing files, perform these steps.
    a. Click **Back** to adjust the settings in the **Globals and Access Paths** pages so that the wizard can locate the missing files.
    b. Repeat till you narrow down the number of missing files.

NOTE

Some old files do not work with the <*target*> implementation, there will be some files missing.

21. Click **Next** .

The **CodeWarrior Project Importer - Project Name** page appears. This page lets you specify the name and select a location for the newly imported project.

22. To specify a name and location to the imported project, perform these steps.
    a. Enter a name for the converted Eclipse project, in the **New Project Name** text box. By default, the old project name is specified.
    b. Check **Use default location** to save the project to the default Eclipse workspace. By default, the location of the project is the directory of the classic project and not the default Eclipse workspace.

Tip

If you want to save the converted project to a location other than the default Eclipse workspace, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project. Alternatively, In the **Location** text box, enter the full path of the directory in which you want to save your project including the project name.

23. Click **Finish** .

    The **CodeWarrior Classic Project** wizard translates the classic CodeWarrior project and the new Eclipse project appears in the **CodeWarrior Project** view of the **Workbench** window.

Chapter 3
Build Properties for Bareboard Projects

This chapter explains build properties for a Microcontrollers project. The Microcontrollers New Bareboard Project wizard uses the information it gathers from you to set up the project's build and launch configurations.

A project's build configuration contains information on the tool settings used to make the program. For example, it describes the compiler and linker settings, and the files involved, such as source and libraries.

A project's launch configuration describes how the IDE starts the program, such as whether it executes by itself on a target, or under debugger control. Launch configurations also specify the core the program executes on (if the target processor has multiple cores). They also specify the connection interface and communications protocol that the debugger uses to control the environment that the program executes in.

NOTE

The settings of the CodeWarrior IDE's build and launch configuration correspond to an object called a target made by the classic CodeWarrior IDE.

When the New Bareboard Project wizard completes its process, it generates launch configurations with names that follow the pattern projectname - configtype - targettype, where:

- projectname represents the name of the project
- configtype represents the project's name, which usually describes the build configuration
- targettype represents the type of target software or hardware on which the launch configuration acts

For each launch configuration, you can specify build properties, such as:

- additional libraries to use for building code
- behavior of the compilers, linkers, assemblers, and other build-related tools
- specific build properties, such as the byte ordering of the generated code

The topics in this chapter are:

3.1 Changing Build Properties

The New Bareboard Project wizard creates a set of build properties for the project. You can modify these build properties to better suit your needs.

Perform these steps to change build properties:

1. Start the IDE.
2. In the **CodeWarrior Projects** view, select the project for which you want to modify the build properties.
3. Select **Project > Properties**.

   The **Properties** window appears. The left side of this window has a properties list. This list shows the build properties that apply to the current project.

4. Expand the **C/C++ Build** property.
5. Select **Settings** .

   The **Properties** window shows the corresponding build properties..

6. Use the Configuration drop-down list to specify the launch configuration for which you want to modify the build properties.
7. Click the **Tool Settings** tab.

   The corresponding page appears.

8. From the list of tools on the **Tool Settings** page, select the tool for which you want to modify properties.
9. Change the settings that appear in the page.

10. Click **Apply** .

    The IDE saves your new settings.

You can select other tool pages and modify their settings. When you finish, click OK to save your changes and close the **Properties** window.

3.2 Restoring Build Properties

If you modify a build configuration that the new project wizard generates, you can restore that configuration to its default state. You might want to restore the build properties in order to have a factory-default configuration, or to revert to a last-known working build configuration. To undo your modifications to build properties, click the **Restore Defaults** button at the bottom of the **Properties** window.

This changes the values of the options to the absolute default of the toolchain. By default, the toolchain options are blank.

For example, when a HCS08 project is created the **Linker > Input** panel has some values set for the **Parameter File** and **Libraries** options, which are specific to the project. Clicking the **Restore Defaults** button defaults the values of the **Parameter File** and **Libraries** options to the blank state of the toolchain.

3.3 Defining C/C++ Build Settings and Behavior

The **C/C++ Build** page includes all builder-specific property pages.

- Define Build Settings
- Define Build Behavior

NOTE

Modifying settings such as the **Generate makefiles automatically** option, might enable or disable some parameters in some situations and change the availability of other property pages.

3.3.1 Define Build Settings

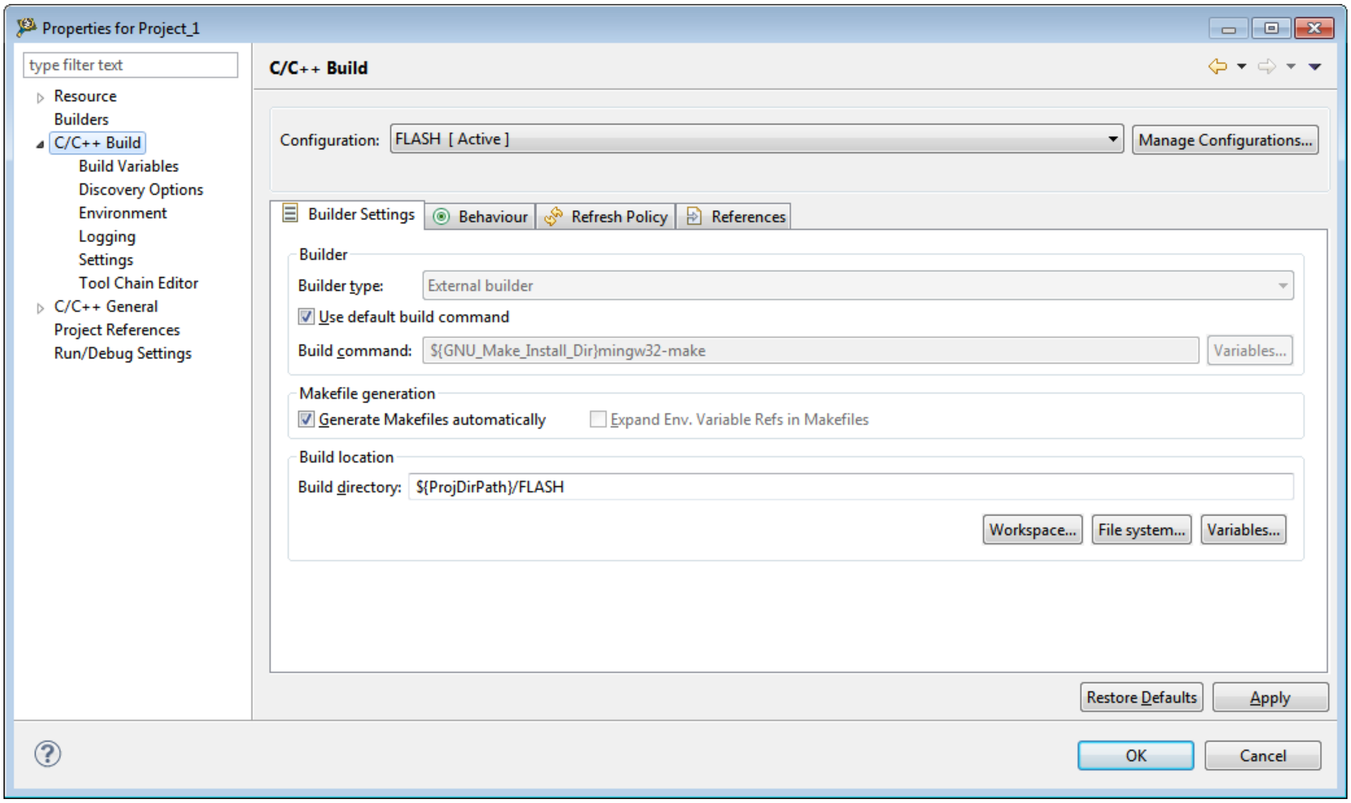
To define build settings, perform these steps.

1. Start the IDE.
2. In the **CodeWarrior Projects** view, select the project for which you want to modify the build settings.
3. Select **Project > Properties**.

   The **Properties for <project>** window appears. The left side of this window has a properties list. This list shows the build properties that apply to the current project.

4. Select **C/C++ Build**.

   The **C/C++ Build** page appears.



Configuration: FLASH [Active]

Builder type: External builder

☑ Use default build command

Build command: ${GNU_Make_Install_Dir}mingw32-make

☑ Generate Makefiles automatically   □ Expand Env. Variable Refs in Makefiles

Build directory: ${ProjDirPath}/FLASH

Figure 3-1. C/C++ Build Page - Builder Settings

5. Click the **Builder Settings** tab.

The builder settings for the selected build configuration appears. The table below describes the builder settings options.

Table 3-1. Builder Settings Options

Group	Option	Description
Build Configuration	Configuration	Specifies the type of configurations for the selected project.
Build Configuration	Manage configurations	Click to open the **Manage Configurations** dialog box that lets you set configurations based on the specified toolchains of the selected project. You can also create new configurations, rename an existing configuration, or remove the ones that are no longer required.
Builder	Builder type	Specifies the type of builder to use: • Internal builder - Builds C/C++ programs using a compiler that implements the C/C++ Language Specifications. • External builder - External tools let you configure and run programs and Ant buildfiles using the Workbench, which can be saved and run at a later time to perform a build.
Builder	Use default build command	Check to indicate that you want to use the default make command. Clear when you want to use a new make command. This option is only available when the Builder type option is set to External.
Builder	Build command	Specifies the default command used to start the build utility for your specific toolchain. Use this field if you want to use a build utility other than the default make command.
Builder	Variables	Click to open the **Select build variable** dialog box and add the desired environment variables and custom variables to the build command.

Table continues on the next page...

Table 3-1. Builder Settings Options (continued)

Group	Option	Description
Makefile generation	Generate Makefiles automatically	Check to enable Eclipse change between two different CDT modes: it either uses the customer's makefile for the build, if one exists, or it generates makefiles for the user.
Makefile generation	Expand Env. Variable Refs in Makefiles	Check to define whether environment variables should be expanded in makefile.
Build location	Build directory	Specifies the location where the build operation takes place. This location will contain the generated artifacts from the build process. This option appears disabled when the Generate Makefiles automatically option is enabled.
Build location	Workspace	Click to open the **Folder Selection**dialog box and select a workspace location for the project. This is the directory that will contain the plug-ins and features to build, including any generated artifacts.
Build location	File system	Click to open the **Browse For Folder** dialog box and select a folder.
Build location	Variables	Click to open the **Select build variable** dialog box and select a variable to specify as an argument for the build directory, or create and configure simple build variables which you can reference in build configurations that support variables.

6. Make the desired changes and click **OK**.

   The **Properties for <project>** window will close.

3.3.2 Define Build Behavior

To define build behavior, perform these steps.

1. Start the IDE.

2. In the **CodeWarrior Projects** view, select the project for which you want to modify the build settings.
3. Select **Project > Properties**.

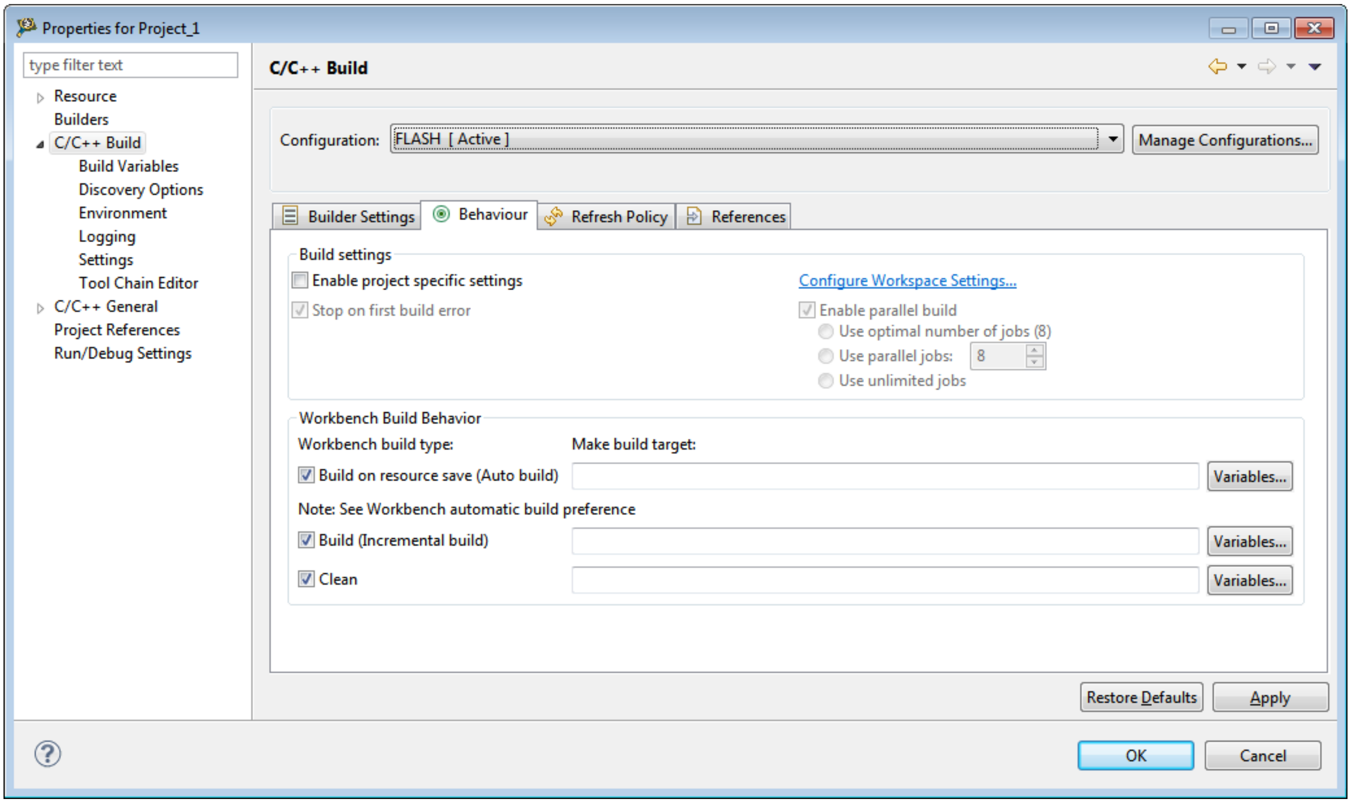
   The **Properties** window appears. The left side of this window has a properties list. This list shows the build properties that apply to the current project.

4. Select **C/C++ Build**.

   The **C/C++ Build** page appears.

5. Click the **Behaviour** tab.

   The behavior settings for the selected build configuration appears.



Configuration: FLASH [Active]

Build settings

☐ Enable project specific settings

☑ Stop on first build error

☑ Enable parallel build

○ Use optimal number of jobs (8)

○ Use parallel jobs: 8

○ Use unlimited jobs

Workbench Build Behavior

Workbench build type: Make build target:

☑ Build on resource save (Auto build)

Note: See Workbench automatic build preference

☑ Build (Incremental build)

☑ Clean

Figure 3-2. C/C++ Build Page - Behaviour

The table below describes the builder settings options.

Table 3-2. Behavior Options

Group	Option	Description
Build settings	Enable project specific settings	Check if you want to enable project specific settings.

Table continues on the next page...

Table 3-2. Behavior Options (continued)

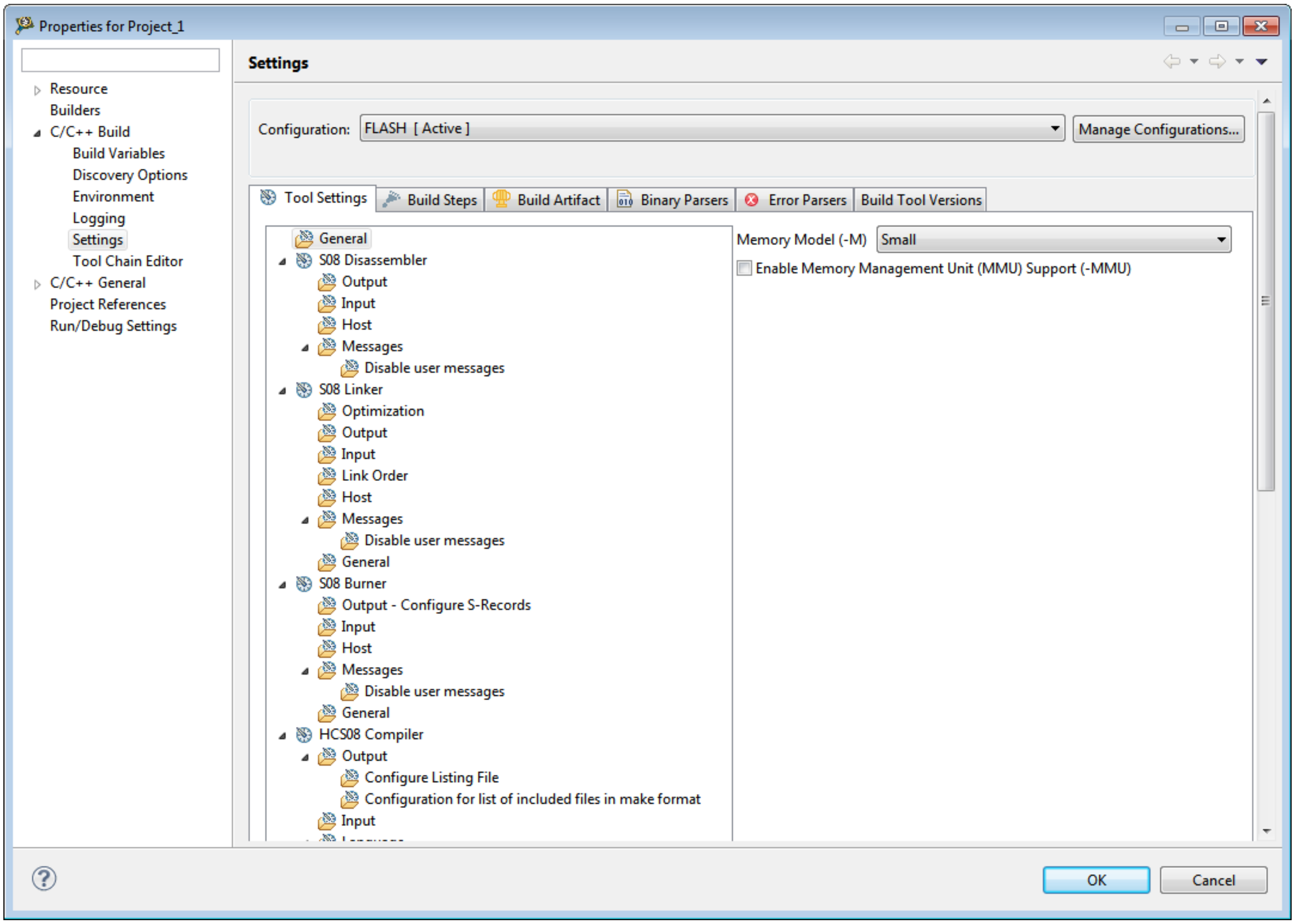
Group	Option	Description
Build settings	Stop on first build error	Check to stop building when Eclipse encounters an error. Clearing this option is helpful for building large projects as it enables make to continue making other independent rules even when one rule fails.
Configure Workspace Settings	Enable parallel build	Check to activate the generation of parallel builds. However, you need to determine the number of parallel jobs to perform: • Use optimal jobs number - Lets the system determine the optimal number of parallel jobs to perform. • Use parallel jobs - Lets you specify the maximum number of parallel jobs to perform. • Use unlimited jobs - Lets the system perform unlimited jobs.
Workbench Build Behavior	Workbench build type	Specifies the builder settings when instructed to build, rebuild, and clean.
Workbench Build Behavior	Build on resource save (Auto build)	Check to build your project whenever resources are saved. By default, this option is selected and builds occur automatically each time resources are modified. Clear if you do want that the build occurs only manually using a menu item.
Workbench Build Behavior	Build (Incremental Build)	Defines what the standard builder will call when an incremental build is performed.
Workbench Build Behavior	Variables	Click to open the **Select build variable** dialog box and add variables to the make build target command.
Workbench Build Behavior	Clean	Defines what the standard builder calls when a clean is performed. The make clean is defined in the makefile.
Workbench Build Behavior	Variables	Click to open the **Select build variable** dialog box and add variables to the make build target command.

6. Make the desired changes and click **OK**.

   The **Properties for <project>** window will close.

3.4 Build Properties for S08

The **Properties for** <*project*> window shows the corresponding build properties for an HCS08 project.



Configuration: FLASH [Active]

Memory Model (-M) Small

☐ Enable Memory Management Unit (MMU) Support (-MMU)

Figure 3-3. Build Properties - S08

The following table lists the build properties specific to developing software for HCS08.

The properties that you specify in the **Tool Settings** panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> dialog box.

Table 3-3. Build Properties for S08

Build Tool	Build Properties Panels
General	General
S08 Disassembler	S08 Disassembler > Output
	S08 Disassembler > Input
	S08 Disassembler > Host
	S08 Disassembler > Messages
	S08 Disassembler > Messages > Disable user messages
S08 Linker	S08 Linker > Optimization
	S08 Linker > Output
	S08 Linker > Input
	S08 Linker > Host
	S08 Linker > Messages
	S08 Linker > Messages > Disable user messages
	S08 Linker > General
S08 Burner	S08 Burner > Output > Configure S-Record
	S08 Burner > Input
	S08 Burner > Host
	S08 Burner > Messages
	S08 Burner > Messages > Disable user messages
	S08 Burner > General
HCS08 Compiler	HCS08 Compiler > Output
	HCS08 Compiler > Output > Configure Listing File
	HCS08 Compiler > Output > Configuration for list of included files in make format
	HCS08 Compiler > Input
HCS08 Compiler	HCS08 Compiler > Language
	HCS08 Compiler > Language > CompactC++ features
	HCS08 Compiler > Host
	HCS08 Compiler > Code Generation
	HCS08 Compiler > Messages
	HCS08 Compiler > Messages > Disable user messages
	HCS08 Compiler > Preprocessor
	HCS08 Compiler > Type Sizes
	HCS08 Compiler > General
	HCS08 Compiler > Optimization
	HCS08 Compiler > Optimization > Tree optimizer
	HCS08 Compiler > Optimization > Optimize Library Function
	HCS08 Compiler > Optimization > Branch Optimizer

Table continues on the next page...

Table 3-3. Build Properties for S08 (continued)

Build Tool	Build Properties Panels
	HCS08 Compiler > Optimization > Peephole Optimization
HCS08 Assembler	HCS08 Assembler > Output
	HCS08 Assembler > Output > Configure listing file
	HCS08 Assembler > Input
	HCS08 Assembler > Language
	HCS08 Assembler > Language > Compatibility modes
	HCS08 Assembler > Host
	HCS08 Assembler > Code Generation
	HCS08 Assembler > Messages
	HCS08 Compiler > Messages > Disable user messages
	HCS08 Assembler > General
HCS08 Preprocessor	HCS08 Preprocessor > Preprocessor Settings

3.4.1 General

Use this panel to specify the memory model that the architecture uses. The build tools (compiler, linker, and assembler) use the properties that you specify.

The following table lists and describes the memory model options for HCS08.

Table 3-4. Tool Settings - General

Option	Description
Memory Model (-M)	Specify the memory model for the build tools: • **Tiny** - Assumes that data pointers have 8-bit addresses unless explicitly specified with the keyword __far • **Small** - Default memory model; assumes that all functions and pointers have 16 bit addresses and requires code and data to be located in 64 kilobytes address space • **Banked** - Lets you place program code into atmost 256 pages of 16 kilobytes each, but does not affect data allocation
Enable Memory Management Unit (MMU) Support (-MMU)	Check to inform the compiler that `CALL` and `RTC` instructions are available, enabling code banking, and that the current architecture has extended data access capabilities, enabling support for `__linear` data types. This option can be used only when `-Cs08` is enabled.

3.4.2 S08 Disassembler

Use this panel to specify the command, options, and expert settings for S08 Disassembler.

The following table lists and describes the Disassembler options.

Table 3-5. Tool Settings - Disassembler Options

Option	Description
Command	Shows the location of the disassembler executable file; default is `${HC08Tools}/decoder.` You can specify additional command line options for the disassembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} -O${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`
Command line pattern	

3.4.2.1 S08 Disassembler > Output

Use this panel to control how the disassembler generates the output file.

The following table lists and describes the output options for S08 disassembler.

Table 3-6. Tool Settings - Disassembler > Output Options

Option	Description
Print full listing	Prints a listing with the header information of the object file.
Write disassembly listing with source code	Check to enable the decoder decoding Freescale object files write the source code within the disassembly listing. This option setting is default for the Freescale object files as input.
Decode DWARF section	Check to write the DWARF section information in the listing file. Decoding from the DWARF section inserts this information in the listing file.
Configure which parts of DWARF information to decode	Check to configure parts of DWARF information to decode.
Decode ELF sections	Check to ensure that the ELF section information is also written to the listing file. Decoding from the ELF section inserts the following information in the listing file.
Dump ELF sections in LST file	Check to generate a HEX dump of all ELF sections in a LST file.
Produce inline assembly file	Check to ensure that the output listing is an inline assembly file without additional information, but in C comments.

Table continues on the next page...

Table 3-6. Tool Settings - Disassembler > Output Options (continued)

Option	Description
No symbols in disassembled listing	Check to prevent symbols from printing in the disassembled listing.
Shows the cycle count for each instruction	Check to ensure that each instruction line contains the count of cycles in '[',']' braces. The cycle count is written before the mnemonics of the instruction. Note that the cycle count display is not supported for all architectures.
Write disassembly listing only	Check to ensure that the Decoder decoding Freescale object files writes the source code within the disassembly listing only.
Write disassembly listing with source and all comments	Check to write the origin source and its comments within the disassembly listing.

3.4.2.2 S08 Disassembler > Input

Use this panel to control how the disassembler generates the input file.

The following table lists and describes the input options for HCS08 disassembler.

Table 3-7. Tool Settings - Disassembler > Input Options

Option	Description
Object File Format	Defines the object file format.
Set processor	Specifies which processor should be decoded. For object files, libraries and applications, the processor is usually detected automatically. For S-Record and Intel Hex files, however, the decoder cannot determine which CPU the code is for, and therefore the processor must be specified with this option to get a disassembly output. Without this option, only the structure of a S-Record file is decoded. The following values are supported: HC08, HC08:HCS08, HC11, HC12, HC12:CPU12, HC12:HCS12, HC12:HCS12X, HC16, M68k, MCORE, PPC, RS08, 8500, 8300, 8051 and XA.

3.4.2.3 S08 Disassembler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-8. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the disassembler uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the disassembler with additional arguments (for example, files and disassembler options). If you start the disassembler with arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the disassembler compiles the files in a batch mode. No disassembler window is visible and the disassembler terminates after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.

Table continues on the next page...

Table 3-8. Tool Settings - Messages Options (continued)

Option	Description
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the disassembler creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.4.2.3.1 S08 Disassembler > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-9. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.4.3 S08 Linker

Use this panel to specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the general, libraries, and search path settings.

The following table lists and describes the linker options for HCS08.

Table 3-10. Tool Settings - Linker Options

Option	Description
Command	Shows the location of the linker executable file. Default value is `"${HC08Tools}/linker.exe"`. You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} -add( ${INPUTS} )`

3.4.3.1 S08 Linker > Optimization

Use this panel to control linker optimizations. The linker's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the linker optimization options for HCS08 **.**

Table 3-11. Tool Settings - Linker > Optimization Options

Option	Description
Allocation over segment boundaries (-Alloc)	The linker supports to allocate objects from one ELF section into different segments. The allocation strategy controls where space for the next object is allocated as soon as the first segment is full. In the AllocNext strategy, the linker always takes the next segment as soon as the current segment is full. Holes generated during this process are not used later. With this strategy, the allocation order corresponds to the definition order in the object files. Objects defined first in a source file are allocated before later defined objects.

Table continues on the next page...

Table 3-11. Tool Settings - Linker > Optimization Options (continued)

Option	Description
	In the AllocFirst strategy, the linker checks for every object, if there is a previously only partially used segment, into which the current object does fit. This strategy does not maintain the definition order. In the AllocChange strategy, the linker checks as soon as a object does no longer fit into the current segment, if there is a previously only partially used segment, into which the current object does fit. This strategy does not maintain the definition order, but it does however use fewer different ranges than the AllocFirst case.
Allocate non referenced overlap variables (-CAllocUnusedOverlap)	When Smart Linking is switched off, defined but unreferenced overlapped variables are not allocated by default. Such variables do not belong to a specific function, therefore they cannot be allocated overlapped with other variables. This option only changes the behavior of variables in the special _OVERLAP segment. This segment is used only to allocate parameters and local variables for processors which do not have a stack. Not allocating an unreferenced overlap variable is similar to not allocating a variable on the stack for other processors. If you use this stack analogy, then allocating such variables this way corresponds to allocating unreferenced stack variables in global memory. This option allows allocation of all defined objects. Using this option is not recommended.
Enable automatic const placement (-ConstDist)	With this option the linker constant optimizer is enabled. Instead of performing usual linking actions, the linker generates a data distribution file which contains optimized distribution for constant objects.
Specify constant distribution segment name (-ConstDistSeg)	When this option is enabled, it's possible to specify the name of the constant distribution segment.
Allocate non specified const segments in RAM (-CRam)	This option allocates constant data segments not explicitly allocated in a READ_ONLY segment in the default READ_WRITE segment. This was the default for old versions of the linker, so this option provides a compatible behavior with old linker versions.
Enable automatic data placement (-DataDist)	With this option the linker data optimizer is enabled. Instead of performing usual linking actions, the linker generates a data distribution file which contains optimized distribution.
Specify data distribution file name (-DataDistFile)	When this option is enabled, it's possible to specify the name of the data distribution file. There, all distributed data and how the compiler has to reallocate them are listed.
Generate data optimizer information file (-DataDistInfo)	When this option is enabled, the data optimizer generates a data distribution information file giving information on object to segment mapping
Specify data distribution segment name (-DataDistSeg)	When this option is enabled, it's possible to specify the name of the data distribution segment.
Enable distribution optimization (-Dist)	This option enables the linker optimizer. Instead of a link, the linker generates a distribution file which contains an optimized distribution.

Table continues on the next page...

Table 3-11. Tool Settings - Linker > Optimization Options (continued)

Option	Description
Specify distribution file name (-DistFile)	Enable this option to specify the name of the distribution file. The distribution file lists all distributed functions and specifies how the compiler reallocates them.
Generate optimizer information file (-DistInfo)	Using this option, the optimizer generates a distribution information file containing a list of all sections and their functions. Available function information includes the old size, optimized size, and new calling convention.
Choose optimizing method (-DistOpti)	Enable this option to choose the optimizing method. With the FillBanks argument the linker minimizes the free space in every bank. FillBanks is most effective for functions using the near calling convention. Use the CodeSize argument to minimize code when free space within the banks is no concern.
Specify distribution segment name (-DistSeg)	Use this option to specify the name of the distribution segment.
Specify library file name (-LibFile)	When this option is enabled,linker generates file<filename> which has information about the current libraries and also about the files with which they should be replaced with.
Enable library option file generation (_LibOptions)	Enables library information generation. When this option is enabled,linker generates file (default libFile.txt) which has information about the current library and the startup file and also about the files with which they should be replaced with.
Specify data optimizer options file name (-OptioneFile)	Specifies the name of the file that contains the set of linker-generated compiler options. When this option is enabled, linker places the second step compiler options in the specified file<filename>.
Enable option file generation (-Options)	Enables compiler option generation. The generated options will be used for second step compilation. Linker generates a text file containing a compiler option for the second step (one of the following: -ConstQualiNear, -NonConstQualiNear, -Mb). The content of the file is appended to the compiler options for the second compilation step.
Specify library file name (-P2LibFileName)	Specifies the name of the library information file. When this option is enabled in second link step,linker reads file<filename> which has information about the libraries.
Enable option to read libFile.txt in P2 (-ReadLibFile)	Instructs the linker to read in the library information file that it generated in step one. This option is passed in second link step. It tells the linker to read library information file(default libFile.txt).
Emit StartUp information to library info file (-StartUpInfo)	The information about the current startup file and the replacement startup file will be added to the library file(default libFile.txt) and used during the second compile-link step.
Overlap constants in ROM (-COCC)	Defines the default if constants and code should be optimized; commands `DO_OVERLAP_CONSTS` and `DO_NOT_OVERLAP_CONSTS` take precedence over the option.
Optimize copy down (-OCopy)	Changes the copy down structure to use few spaces. The optimization does assume that the application does perform both the zero out and the copy down step of the global initialization. If a value is set to zero by the zero out, then zero

Table 3-11. Tool Settings - Linker > Optimization Options

Option	Description
	values are removed from the copy down information. The resulting initialization is not changed by this optimization if the default startup code is used.

3.4.3.2 S08 Linker > Output

Use this panel to control how the linker formats the listing file, as well as error and warning messages.

The following table lists and describes the linker output options for HCS08.

Table 3-12. Tool Settings - Linker > Output Options

Option	Description
Link as ROM library (-AsROMlib)	Check to link the application as a ROM library. This option has the same effect as specifying AS ROM_LIB in the linker parameter file.
Generate S_record file (-B)	Check to specify that in addition to an absolute file, also an S-record file should be generated. The name of the srecord file is the same as the name of the abs file, except that the extension "SX" is used. The default.env variable "SRECORD" may specify an alternative extension.
Check if objects overlap in the absolute file (even if different address spaces) (-CheckAcrossAddrSpace)	Check to instruct the linker to check if objects overlap, taking into account their address space.
Define the default value of the EPAGE register (-DefaultEpage)	Defines the reset value for the EEPROM Page Index Register (EPAGE). The value is specific to the actual S12(X) derivative.
Define the default value of the PPAGE register (-DefaultPpage)	Defines the reset value for the Program Page Index Register (PPAGE). The value is specific to the actual S12(X) derivative.
Define the default value of the RPAGE register (-DefaultRpage)	Defines the reset value for the RAM Page Index Register (RPAGE). The value is specific to the actual S12(X) derivative.
Generate map file (-M)	Check to scan source files for dependencies and emit a mapfile, without generating object code.
Mapping for memory space 0x4000-0x7FFF	This option sets the memory mapping for addresses between 0x4000 and 0x7FFF. This mapping is determined by the MMC control register (the ROMHM and RAMHM bits) and the compiler must be aware of the current setting to correctly perform address translations.
Never check section qualifier compatibility (-NoSectCompat)	For some target CPU's, when placing a section in a segment the linker checks if the qualifiers of the section are compatible with the ones of the segment (for instance when placing .text into RAM may result in a linker error).This option disables the check.

Table continues on the next page...

Table 3-12. Tool Settings - Linker > Output Options (continued)

Option	Description
Strip symbolic information (-S)	Check to disable the generation of DWARF sections in the absolute file to save memory space.
Generate fixups in abs file (-SFixups)	Check to ensure compatibility with previous linker versions. Usually, absolute files do not contain any fixups because all fixups are evaluated at link time. But with fixups, the decoder might symbolically decode the content in absolute files. Some debuggers do not load absolute files which contain fixups because they assume that these fixups are not yet evaluated. But the fixups inserted with this option are actually already handled by this linker.
Enable Stack Consumption Computation (-StackConsumption)	The linker computes maximum stack effect for given application when the option is enabled and places the result in the output .map file.
Specify statistic file (e.g. statistic.txt) (-StatF)	Specify the name of the linker statistic file. The statistic file reports each allocated object and its attributes. Every attribute is separated by a tab character, so it can be easily imported into a spreadsheet/database program for further processing.

3.4.3.3 S08 Linker > Input

Use this panel to specify the parameter file path, startup function, object file search paths, and any additional libraries that the C/C++ Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The IDE first looks for an include file in the current directory, or the directory that you specify in the `INCLUDE` directive. If the IDE does not find the file, it continues searching the paths shown in this panel. The IDE keeps searching paths until it finds the `#include` file or finishes searching the last path at the bottom of the Include File Search Paths list. The IDE appends to each path the string that you specify in the `INCLUDE` directive.

NOTE

> The IDE displays an error message if a header file is in a different directory from the referencing source file. Sometimes, the IDE also displays an error message if a header file is in the same directory as the referencing source file.

For example, if you see the message `Could not open source file myfile.h`, you must add the path for `myfile.h` to this panel.

The following table lists and describes the linker input options for HCS08.

Table 3-13. Tool Settings - Linker > Input Options

Option	Description
Parameter File	Shows the path of the parameter file. Default value is ${ProjDirPath}/Project_Settings/Linker_Files/Project.prm.
Specify startup function (-E)	Defines the application entry point.
Search paths (-L)	Shows the list of all search paths; the ELF part of the linker searches object files first in all paths and then the usual environment variables are considered.
Libraries	Lists paths to additional libraries that the C/C++ linker uses. Default value is "${MCUToolsBaseDir}/lib/hc08c/lib/ansiis.lib"
Link case insensitive	With this option, the linker ignores object name capitalization. This option supports case-insensitive linking of assembly modules. Since all identifiers are linked case insensitive, this also affects C or C++ modules. This option only affects the comparison of names of linked objects. Section names or the parsing of the link parameter file are unaffected. They remain case sensitive.
Object File Format	Defines the object file format.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object file search paths.

Table 3-14. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the Add directory path dialog box.

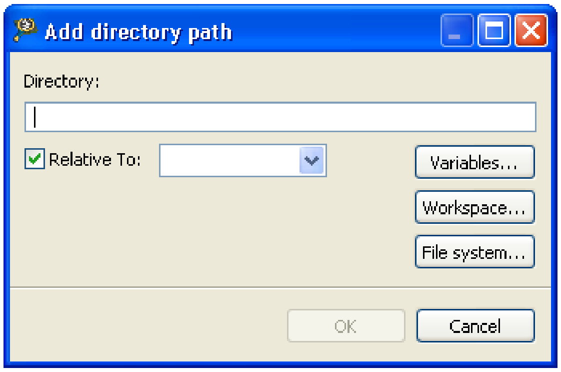
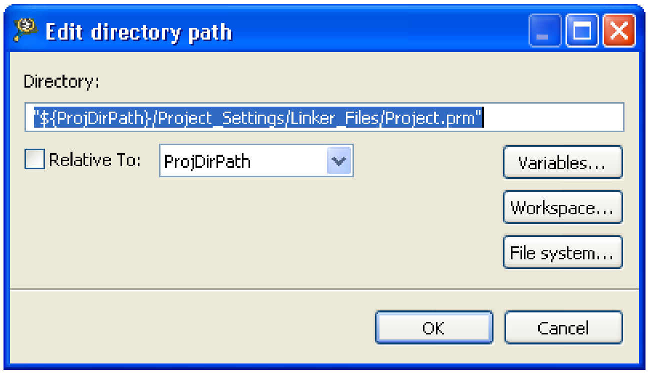


Figure 3-4. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



Directory: "${ProjDirPath}/Project_Settings/Linker_Files/Project.prm"

□ Relative To: ProjDirPath

Figure 3-5. Edit directory path Dialog Box

The buttons in the **Add directory path and Edit directory path** dialog boxes help work with the object file search paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the object file search path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object files.

Table 3-15. Libraries Toolbar Buttons

Button	Description
+	Add - Click to open the **Add file path** dialog box and specify location of the library you want to add.

Table continues on the next page...

Table 3-15. Libraries Toolbar Buttons (continued)

Button	Description
	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
	Edit - Click to open the **Edit file path** dialog box and update the selected path.
	Move up - Click to move the selected path one position higher in the list.
	Move down - Click to move the selected path one position lower in the list.

The following figure shows the Add file path dialog box.

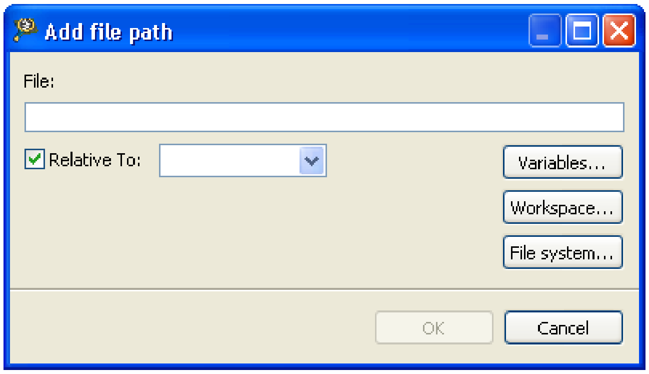
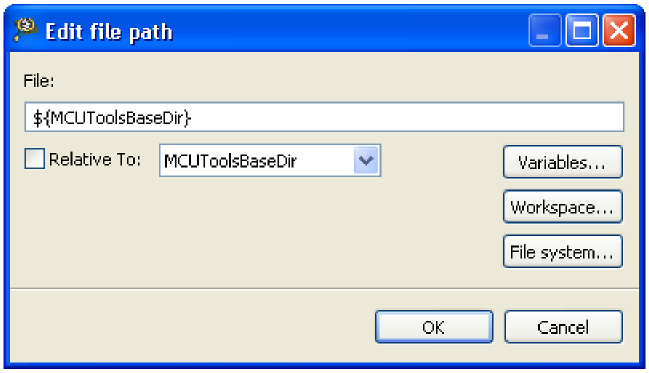


Figure 3-6. Tool Settings - Linker > Libraries - Add file path Dialog Box

The following figure shows the Edit file path dialog box.



File: ${MCUToolsBaseDir}

□ Relative To: MCUToolsBaseDir

Figure 3-7. Tool Settings - Linker > Libraries - Edit file path Dialog Box

The buttons in the **Add file path and Edit file path** dialog boxes help work with the file paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

- Workspace **-** Click to display the **File Selection** dialog box and specify the file path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Open** dialog box and specify the file path. The resulting absolute path appears in the appropriate list.

3.4.3.4 S08 Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-16. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.4.3.5 S08 Linker > Host

Use this panel to specify the host settings of the HCS08.

The following table lists and describes the memory model options for HCS08.

Table 3-17. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link

Table 3-17. Tool Settings - Host

Option	Description
	a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.4.3.6 S08 Linker > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-18. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Linker uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Linker with additional arguments (for example, files and Linker). If you start the Linker arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the Linker the files in a batch mode. No Linker is visible and the Linker after job completion.

Table continues on the next page...

Table 3-18. Tool Settings - Messages Options (continued)

Option	Description
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m)(-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Linker creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.4.3.6.1 S08 Linker > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-19. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.4.3.7 S08 Linker > General

Use this panel to specify the general linker behavior.

The following table lists and describes the **general linker options for HCS08.**

Table 3-20. Tool Settings - Linker > General Options

Option	Description
Other flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI. Default value is: `-WmsgSd1100 -WmsgSd1912`

3.4.4 S08 Burner

Use the Burner for HCS08 Preference Panel to map *.bbl (batch burner language) files to the Burner Plug-In. When the project folder contains a *.bbl file, *.bbl file processing during the post-link phase uses the settings in the Burner preference panel.

The following table lists and describes the burner options for HCS08.

Table 3-21. Tool Settings - Burner Options

Option	Description
Command	Shows the location of the burner executable file. Default value is: `"${HC08Tools}/burner"`. You can specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the burner will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${INPUTS}`.

3.4.4.1 S08 Burner > Output > Configure S-Record

Use this panel to control how the burner generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the general options for HCS08 configure S-Record.

Table 3-22. Tool Settings - Burner > Output > Configure S-Record Options

Option	Description
Disable all (-Ns)	Disables generation of all start (S0) and end records (S7, S8, or S9)
No path in S0-record	Removes the path (if present) from the file name in the S0 record
No S9-record	Disables generation of S9-record
No S8-record	Disables generation of S8-record
No S7-record	Disables generation of S7-record
No S0-record	Disables generation of S0-record

3.4.4.2 S08 Burner > Input

Use this panel to specify the execute command file of the Burner input.

The following table lists and describes the inputs options for burner.

Table 3-23. Tool Settings - Burner > Input Options

Option	Description
Execute command file	This option causes the Burner to execute a Batch Burner command file (usual extension is .bbl).

3.4.4.3 S08 Burner > Host

Use this panel to specify the host settings of the HCS08.

The following table lists and describes the memory model options for HCS08.

Table 3-24. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.4.4.4 S08 Burner > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-25. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Burner uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Burner with additional arguments (for example, files and Burner options). If you start the Burner with arguments (for example, from the Make Tool or with the ` %f' argument from the CodeWright IDE), the Burner compiles the files in a batch mode. No Burner window is visible and the Burner terminates after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.

Table continues on the next page...

Table 3-25. Tool Settings - Messages Options (continued)

Option	Description
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Burner creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.4.4.4.1 S08 Burner > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-26. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.4.4.5 S08 Burner > General

Use this panel to specify the general linker behavior.

The following table lists and describes the general burner options for **HCS08.**

Table 3-27. Tool Settings - Burner > General Options

Option	Description
Other flags	Specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.

3.4.5 HCS08 Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the HCS08 Compiler tree control includes the general and the file search path settings.

The following table lists and describes the compiler options for HCS08.

Table 3-28. Tool Settings - Compiler Options

Option	Description
Command	Shows the location of the compiler executable file. Default value is: `"${HC08Tools}/chc08.exe"`. You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the compiler will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS}${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.

3.4.5.1 HCS08 Compiler > Output

Use this panel to control how the compiler generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the output options for HCS08 compiler.

Table 3-29. Tool Settings - HCS08 Compiler > Output Options

Option	Description
Allocate CONST objects in ROM (-Cc)	Check to enables the Compiler assign const objects into the ROM_VAR segment, which the parameter file assigns to a ROM section.
Encrypt Files (e.g. %f.e%e)(-Eencrypt)	Encrypts using the given key with the -Ekey: Encryption Key option.
Encryption key (-Ekey)	Encrypt files with the given key number (-Eencrypt option).The default encryption key is 0. Using this default is not recommended.
Object File Format	Defines the object file format.
Generate Assembler Include File (e.g. %f.inc)(-La)	Enables the Compiler to generate an assembler include file when the CREATE_ASM_LISTING pragma occurs. The name of the created file is specified by this option. If no name is specified, a default of %f.inc is taken. To put the file into the directory specified by the TEXTPATH: Text File Path environment variable, use the option -la=%n.inc. The %f option already contains the path of the source file. When %f is used, the generated file is in the same directory as the source file. The content of all modifiers refers to the main input file and not to the actual header file. The main input file is the one specified on the command line.
Generate Listing File (e.g. %n.lst)(-Lasm)	Enables the Compiler to generate an assembler listing file directly. The Compiler also prints all assembler-generated instructions to this file. The option specifies the name of the file. If no name is specified, the Compiler takes a default of %n.lst. If the resulting filename contains no path information the Compiler uses the TEXTPATH: Text File Path environment variable. The syntax does not always conform with the inline assembler or the assembler syntax. Therefore, use this option only to review the generated code. It cannot currently be used to generate a file for assembly.
Log predefined defines to file (e.g. predef.h)(-Ldf)	Enables the Compiler to generate a text file that contains a list of the compiler-defined #define. The default filename is predef.h, but may be changed (e.g., -Ldf="myfile.h"). The file is generated in the directory specified by the TEXTPATH: Text File Path environment variable. The defines written to this file depend on the actual Compiler option settings (e.g., type size settings or ANSI compliance). **Note:** The defines specified by the command line (-D: Macro Definition option) are not included. This option may be very useful for SQA. With this option it is possible to document every #define which was used to compile all sources. Note: This option only has an effect if a file is compiled. This option is unusable if you are not compiling a file.

Table continues on the next page...

Table 3-29. Tool Settings - HCS08 Compiler > Output Options (continued)

Option	Description
List of included files to `.inc' file (-Li)	Enables the Compiler to generate a text file which contains a list of the #include files specified in the source. This text file shares the same name as the source file but with the extension, *.inc. The files are stored in the path specified by the TEXTPATH: Text File Path environment variable. The generated file may be used in make files.
Write static output to file (e.g. logfile.txt)(-Ll)	Enables the Compiler append statistical information about the compilation session to the specified file. The information includes Compiler options, code size (in bytes), stack usage (in bytes) and compilation time (in seconds) for each procedure of the compiled file. The Compiler appends the information to the specified filename (or the file make.txt, if no argument given). Set the TEXTPATH: Text File Path environment variable to store the file into the path specified by the environment variable. Otherwise the Compiler stores the file in the current directory.
List of included files in make format (e.g. make.txt)(-Lm)	This option causes the Compiler to generate a text file which contains a list of the #include files specified in the source. The generated list is in a make format. The -Lm option is useful when creating make files. The output from several source files may be copied and grouped into one make file. The generated list is in the make format. The filename does not include the path. After each entry, an empty line is added. The information is appended to the specified filename (or the make.txt file, if no argument is given).
Append object file name to list (e.g.obklist.txt)(-Lo)	This option causes the Compiler to append the object filename to the list in the specified file.The information is appended to the specified filename (or the file make.txt file, if no argument given).
Processor output (e.g. %n.pre)(-Lp)	This option causes the Compiler to generate a text file which contains the preprocessor's output. If no filename is specified, the text file shares the same name as the source file but with the extension, *.PRE (%n.pre). The TEXTPATH environment variable is used to store the preprocessor file.
Strip path information (-NoPath)	Check to enable the compiler remove both unreferenced path reference from your program. This reduces your program's memory footprint.

3.4.5.1.1 HCS08 Compiler > Output > Configure Listing File

Use this panel to configure the listing file options of the HCS08 compiler.

The following table lists and describes the **Configure Listing File** options for the HC(S)08 Compiler.

Table 3-30. Tool Settings - HCS08 Compiler > Output > Configure Listing File Options

Option	Description
Disable all (-Lasmc)	This option configures the output format of the listing file generated with the Generate Listing File option. The addresses, the hex bytes, and the instructions are selectively switched off.
Do not write cycle information (-Lasmc=y)	This option switches off the cycle information from the output format of the listing file.
Do not write the compiler version (-Lasmc=v)	This option switches off the compiler version from the output format of the listing file.
Do not write the source code (-Lasmc=s)	This option switches off the source code from the output format of the listing file.
Do not write the source prolog (-Lasmc=p)	This option switches off the source prolog from the output format of the listing file.
Do not write the instruction (-Lasmc=i)	This option switches off the instruction from the output format of the listing file.
Do not write the function header (-Lasmc=h)	This option switches off the function header from the output format of the listing file.
Do not write the source epilog (-Lasmc=e)	This option switches off the source epilog from the output format of the listing file.
Do not write the code (-Lasmc=c)	This option switches off the code from the output format of the listing file.
Do not write the address (-Lasmc=a)	This option switches off the address from the output format of the listing file.

3.4.5.1.2 HCS08 Compiler > Output > Configuration for list of included files in make format

Use this panel to configure the list of included files in make format of the HCS08 compiler.

The following table lists and describes the **Configuration for list of included files in make format options** for HC(S)08 compiler.

Table 3-31. Tool Settings - HCS08 Compiler > Output > Configuration for List of Included Files in Make Format Options

Option	Description
Disale all (-LmCfg)	This option is used when configuring the List of Included Files in Make Format (-Lm) option. The -LmCfg option is operative only if the -Lm option is also used. The -Lm option produces the `dependency' information for a make file.

Table continues on the next page...

Table 3-31. Tool Settings - HCS08 Compiler > Output > Configuration for List of Included Files in Make Format Options (continued)

Option	Description
Unix style paths (-LmCfg=x)	Use this option to writes the path names in Unix style.
Update information (-LmCfg=u)	This option updates the information in the output file. If the file does not exist, the Compiler creates the file. If the file exists and the current information is not yet in the file, the Compiler appends the information to the file. If the information is already present, the Compiler updates the information. This allows you to specify this suboption for each compilation ensuring that the make dependency file is always up to date.
Write path of object file (-LmCfg=o)	This option writes the full name of the target object file.
Write path of main file (-LmCfg=m)	This option writes the full path of the compiled file. This is necessary when there are files with the same name in different directories.
Use line continuation (-LmCfg=l)	This option uses line continuation for each single entry in the dependency list. This improves readability.
Write path of included files (-LmCfg=i)	This option writes the full path of all included files in the dependency list.

3.4.5.2 HCS08 Compiler > Input

Use this panel to specify file search paths and any additional include files the HCS08 Compiler should use. You can specify multiple search paths and the order in which you want to perform the search.

The IDE first looks for an include file in the current directory, or the directory that you specify in the `INCLUDE` directive. If the IDE does not find the file, it continues searching the paths shown in this panel. The IDE keeps searching paths until it finds the `#include` file or finishes searching the last path at the bottom of the Include File Search Paths list. The IDE appends to each path the string that you specify in the `INCLUDE` directive.

NOTE

The IDE displays an error message if a header file is in a different directory from the referencing source file. Sometimes, the IDE also displays an error message if a header file is in the same directory as the referencing source file.

For example, if you see the message `Could not open source file myfile.h`, you must add the path for `myfile.h` to this panel.

The following table lists and describes the input options for HCS08 Compiler.

Table 3-32. Tool Settings - HCS08 Compiler > Input Options

Option	Description
Filenames are clipped to DOS length (-!)	The filenames are clipped to DOS length (eight characters), when compiling files from MS-DOS file system.
Include File Path (-I)	Specify, delete, or rearrange file search paths.
Recursive Include File Path (-Ir)	Appends a recursive access path to the current #include list. This command is global. **Syntax**`-ir pathpath` The recursive access path to append.
Additional Include Files (-AddInd)	Specify, delete, or rearrange paths to search any additional #include files.
Include files only once (-Pio)	Check to include every header file only once; duplicates are ignored.

The following table lists and describes the toolbar buttons that help work with the file paths.

Table 3-33. Include File Path (-I) Toolbar Buttons

Button	Description
➕	Add - Click to open the Add directory path dialog box and specify location of the library you want to add.
✖	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected path.
⇧	Move up - Click to move the selected path one position higher in the list.
⇩	Move down - Click to move the selected path one position lower in the list.

The following table lists and describes the toolbar buttons that help work with the search paths.

Table 3-34. Additional Include Files (-AddIncl) Toolbar Buttons

Button	Description
➕	Add - Click to open the Add directory path dialog box and specify location of the library you want to add.
✖	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected path.
⇧	Move up - Click to move the selected path one position higher in the list.

Table continues on the next page...

Table 3-34. Additional Include Files (-AddIncl) Toolbar Buttons (continued)

Button	Description
⇩	Move down - Click to move the selected path one position lower in the list.

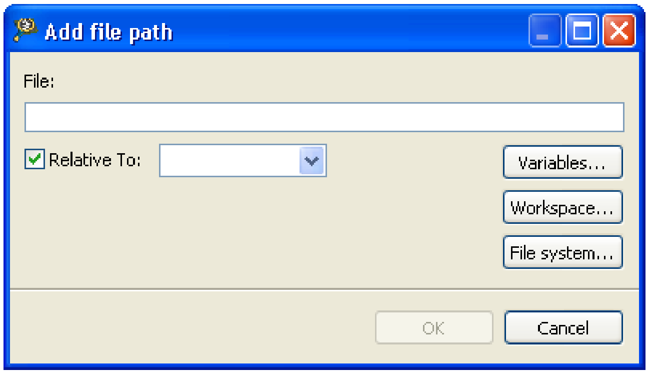
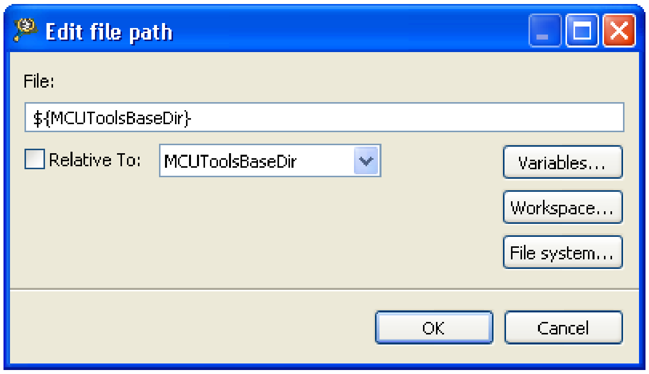


Figure 3-8. Tool Settings - HCS08 Compiler > Input - Add file path Dialog Box



File: ${MCUToolsBaseDir}

□ Relative To: MCUToolsBaseDir

Figure 3-9. Tool Settings - HCS08 Compiler > Input - Edit file path Dialog Box

The buttons in the **Add file path and Edit file path** dialog boxes help work with the paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **File Selection** dialog box and specify the path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Open** dialog box and specify the path. The resulting path appears in the appropriate list.

3.4.5.3 HCS08 Compiler > Language

Use this panel to specify code- and symbol-generation options for the HCS08 Compiler.

The following table lists and describes the language options for HCS08.

Table 3-35. Tool Settings - HCS08 Compiler > Language Options

Option	Description
Strict ANSI (-Ansi)	Check if you want the C compiler to operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ISO specification to all input files. This setting is equivalent to specifying the `- ansi` command-line option. The compiler issues a warning for each ANSI/ISO extension it finds.
C++	With this option enabled, the Compiler behaves as a C++ Compiler. You can select between three different types of C++: • **Full C++ (-C++f) -** Supports the whole C++ language. • **Embedded C++ (-C++e)** - Supports a constant subset of the C++ language. EC++ does not support inefficient things like templates, multiple inheritance, virtual base classes and exception handling. • **CompactC++ (-C++c) -** Supports a configurable subset of the C++ language. You can configure this subset with the option -Cn. • **No C++** - If the option is not set, the Compiler behaves as an ANSI-C Compiler. If the option is enabled and the source file name extension is *.c, the Compiler behaves as a C++ Compiler. If the option is not set, but the source filename extension is .cpp or .cxx, the Compiler behaves as if the -C++f option were set.
Cosmic compatibility mode for space modifiers @near, @far, and @tiny (-Ccx)	Check to allow Cosmic style @near, @far and @tiny space modifiers as well as @interrupt in your C code. The -ANSI option must be switched off. It is not necessary to remove the Cosmic space modifiers from your application code. There is no need to place the objects to sections addressable by the Cosmic space modifiers. The following is done when a Cosmic modifier is parsed: The objects declared with the space modifier are always allocated in a special Cosmic compatibility (_CX) section (regardless of which section pragma is set) depending on the space modifier, on the const qualifier or if it is a function or a variable. Space modifiers on the left hand side of a pointer declaration specify the pointer type and pointer size, depending on the target.
Bigraph and trigraph support (-Ci)	Check to replace certain unavailable tokens with the equivalent keywords.
C++ comments in ANSI-C (-Cppc)	Check to allow C++ comments.
Propagate const and volatile qualifiers for structs (-Cq)	Check to propagate const and volatile qualifiers for structures. If all members of a structure are constant or volatile, the structure itself is constant or volatile. If the structure is declared as constant or volatile, all its members are constant or volatile, respectively.
Conversion from `const T*' to `T*' (-Ec)	Check to enable this non-ANSI compliant extension allows the compiler to treat a pointer to a constant type like a pointer to the non-constant equivalent of the type. Earlier Compilers did not check a store to a constant object through a pointer. This option is useful when compiling older source code.

Table continues on the next page...

Table 3-35. Tool Settings - HCS08 Compiler > Language Options (continued)

Option	Description
Do not pre-process escape sequences in strings with absolute DOS path (-Pe)	When the -Pe option is enabled, the Compiler handles strings in include directives differently from other strings. Escape sequences in include directive strings are not evaluated. The following example: `#include "C:\names.h"` results in exactly the same include filename as in the source file (" `C:\names.h`"). If the filename appears in a macro, the Compiler does not distinguish between filename usage and normal string usage with escape sequence.

NOTE

For more information about the `-Pe` option, refer **Microcontrollers V10.x HC08 Build Tools Reference Manual.**

3.4.5.3.1 HCS08 Compiler > Language > CompactC++ features

Use this panel to select compact C++ features of HCS08 compiler.

The following table lists and describes the compactC++ options for HCS08.

Table 3-36. Tool Settings - HCS08 Compiler > Language > CompactC++ Features Options

Option	Description
Disable all compactC++ features (-Cn)	If the -C++ option is enabled, you can disable the compactC+ + features. • Vf : Virtual functions are not allowed. Avoid having virtual tables that consume a lot of memory. • Tpl : Templates are not allowed. Avoid having many generated functions perform similar operations. • Ptm : Pointer to member not allowed. Avoid having pointer-to-member objects that consume a lot of memory. • Mih : Multiple inheritance is not allowed. Avoid having complex class hierarchies. Because virtual base classes are logical only when used with multiple inheritance, they are also not allowed. • Ctr : The C++ Compiler can generate several kinds of functions, if necessary: - Default Constructor - Copy Constructor - Destructor

Table continues on the next page...

Table 3-36. Tool Settings - HCS08 Compiler > Language > CompactC++ Features Options (continued)

Option	Description
	- Assignment operator With this option enabled, the Compiler does not create those functions. This is useful when compiling C sources with the C++ Compiler, assuming you do not want C structures to acquire member functions. • Cpr : Class parameters and class returns are not allowed. Avoid overhead with Copy Constructor and Destructor calls when passing parameters, and passing return values of class type.
Do not allow virtual functions (-Cn=Vf)	Virtual functions are not allowed. Avoid having virtual tables that consume a lot of memory
Do not allow templates (-Cn=Tpl)	Templates are not allowed. Avoid having many generated functions perform similar operations.
Do not allow pointer to member (-Cn=Ptm)	Pointer to member not allowed. Avoid having pointer-to-member objects that consume a lot of memory.
Do not allow multiple inheritance and virtual base classes (-Cn=Mih)	Multiple inheritance is not allowed. Avoid having complex class hierarchies. Because virtual base classes are logical only when used with multiple inheritance, they are also not allowed.
Do not create compiler defined functions (-Cn=Ctr)	The C++ Compiler can generate several kinds of functions, if necessary: • Default Constructor • Copy Constructor • Destructor • Assignment operator With this option enabled, the Compiler does not create those functions. This is useful when compiling C sources with the C ++ Compiler, assuming you do not want C structures to acquire member functions.
Do not allow class parameters and class returns (-Cn=Ctr)	Class parameters and class returns are not allowed. Avoid overhead with Copy Constructor and Destructor calls when passing parameters, and passing return values of class type.

3.4.5.4 HCS08 Compiler > Host

Use this panel to specify the host settings of the HCS08.

The following table lists and describes the memory model options for HCS08.

Table 3-37. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.4.5.5 HCS08 Compiler > Code Generation

Use this panel to specify code- and symbol-generation options for the HCS08 Compiler

The following table lists and describes the code generation options for HCS08 compiler.

Table 3-38. Tool Settings - HCS08 Compiler > Code Generation Options

Option	Description
Bit field Byte allocation (-BfaB[MS|LS])	By default, bit allocation in byte bitfields proceeds from the least significant bit to the most significant bit. This produces less code overhead in the case of partially- allocated byte bitfields. Options are: • MS: Most significant bit in byte first (left to right) • LS: Least significant bit in byte first (right to left)

Table continues on the next page...

Table 3-38. Tool Settings - HCS08 Compiler > Code Generation Options (continued)

Option	Description
Bit field gap limit (-BfaGapLimitBits)	Check to affect the maximum allowable number of gap bits. The bitfield allocation tries to avoid crossing a byte boundary whenever possible. To optimize accesses, the compiler may insert some padding or gap bits.
Bit field type size reduction	This option is configurable whether or not the compiler uses type-size reduction for bitfields. Type-size reduction means that the compiler can reduce the type of an int bitfield to a char bitfield if it fits into a character. This allows the compiler to allocate memory only for one byte instead of for an integer. Options are: • **Enabled** (-BfaTSRON) • **Disabled** (-BfaTSROFF)
Maximum load factor for switch tables (-100) (-CswMaxLF)	Allows changing the default strategy of the Compiler to use tables for switch statements; is only available if the compiler supports switch tables.
Minimum number of labels for switch tables (-CswMinLB)	Allows changing the default strategy of the Compiler using tables for switch statements; is only available if the compiler supports switch tables.
Minimum load factor for switch tables (100) (-CswMinLF)	Allows the Compiler to use tables for switch statements; is only available if the compiler supports switch tables.
Minimum number of labels for switch search tables (-CswMinSLB)	Allows the Compiler to use tables for switch statements. Using a search table improves code density, but the execution time increases. Every time an entry in a search table must be found, all previous entries must be checked first. For a dense table, the right offset is computed and accessed. In addition, note that all backends implement search tables (if at all) by using a complex runtime routine. This may make debugging more complex.
Switch off code generation (-Cx)	Disables the code generation process of the Compiler. No object code is generated, though the Compiler performs a syntactical check of the source code. This allows a quick test if the Compiler accepts the source without errors.
Do not use CLR for volatile variables in the direct page (-NoClrVol)	Inhibits the use of CLR for volatile variables in the direct page. The CLR instruction on HC08 has a read cycle. This may lead to unwanted lateral effects (e.g. if the variable is mapped over a hardware register).
Qualifier for virtual table pointers (-Qvtp)	Using a virtual function in C++ requires an additional pointer to virtual function tables. The Compiler cannot access the pointer and generates the pointer in every class object when virtual function tables are associated. Options are: • None • Near • Far
Use IEEE32 for double (default is IEEE64)	Check to use doubles that are in IEEE32 instead of IEEE64 (default).
Assume HLI code saves modified registers	With this option set, the compiler assumes that registers touched in HLI are saved or restored in the HLI code as well. If this option is not set, the compiler saves or restores the H, X, and A registers.

3.4.5.6 HCS08 Compiler > Messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-39. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Compiler uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Compiler with additional arguments (for example, files and Compiler options). If you start the Compiler with arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the Compiler compiles the files in a batch mode. No Compiler window is visible and the Compiler terminates after job completion.
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Compiler creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Error for Implicit parameter declaration (-Wpd)	This option prompts the Compiler to issue an ERROR message instead of a WARNING message when the Compiler encounters an implicit declaration. This occurs if the Compiler does not have a prototype for the called function. This option helps prevent parameter-passing errors, which can only be detected at runtime. It requires prototyping each

Table continues on the next page...

Table 3-39. Tool Settings - Messages Options (continued)

Option	Description
	called function before use. Correct ANSI behavior assumes that parameters are correct for the stated call. This option is the same as using - `WmsgSe1801`.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.4.5.6.1 HCS08 Compiler > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-40. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.4.5.7 HCS08 Compiler > Preprocessor

Use this panel to specify preprocessor behavior. You can specify the file paths and define macros.

The following table lists and describes the preprocessor options for HCS08 Compiler.

Table 3-41. Tool Settings - HCS08 Compiler > Preprocessor Option

Option	Description
Define preprocessor macros (-D)	Define, delete, or rearrange preprocessor macros. You can specify multiple macros and change the order in which the IDE uses the macros. Define preprocessor macros and optionally assign their values. This setting is equivalent to specifying the `-D name[=value]` command-line option. To assign a value, use the equal sign (=) with no white space. For example, this syntax defines a preprocessor value named `EXTENDED_FEATURE` and assigns `ON` as its value: `EXTENDED_FEATURE=ON` Note: If you do not assign a value to the macro, the shell assigns a default value of 1.

The following table lists and describes the toolbar buttons that help work with preprocessor macro definitions.

Table 3-42. Define Preprocessor Macros Toolbar Buttons

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the path/macro.
✖	Delete - Click to delete the selected path/macro. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected path/macro.
⇧	Move up - Click to move the selected path/macro one position higher in the list.
⇩	Move down - Click to move the selected path/macro one position lower in the list.

The following figure shows the Enter Value dialog box.

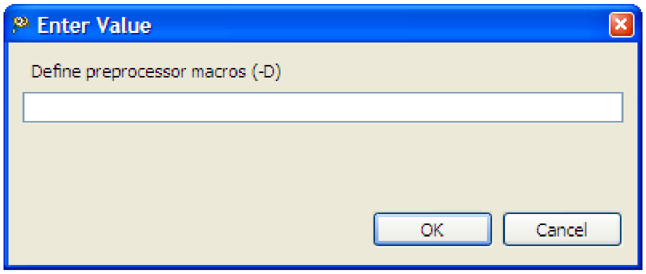
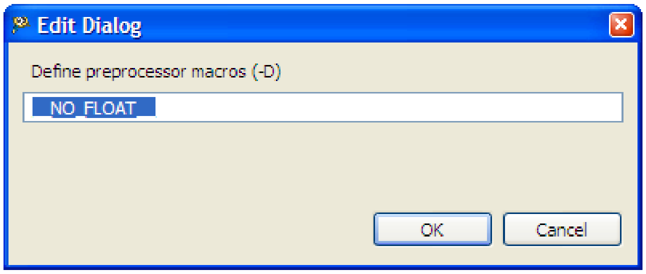


Figure 3-10. Tool Settings - HCS08 Compiler > Preprocessor - Enter Value Dialog Box

The following figure shows the Edit Dialog dialog box.



Define preprocessor macros (-D)

NO_FLOAT

Figure 3-11. Tool Settings - HCS08 Compiler > Preprocessor - Edit Dialog Box

The buttons in the **Enter Value and Edit** dialog boxes help work with the preprocessor macros.

- **OK -** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.4.5.8 HCS08 Compiler > Type Sizes

Use this panel to specify the available data type size options for the HCS08 Compiler.

The following table lists and describes the possible type size options for HCS08 Compiler using the `-T` option.

Table 3-43. Tool Settings - HCS08 Compiler > Type Sizes

Option	Description
char	Selects the size of the char type. Options are: • Default (unsigned 8bit) • unsigned 8bit (-TuCC1) • signed 8bit (-TsCC1) • signed 16bit (-TsCC2) • signed 32bit (-TsCC4)
short	Selects the size of the short type. Options are: • Default (16bit) • signed 8bit (-TS1) • signed 16bit (-TS2) • signed 32bit (-TS4)
int	Selects the size of the int type. Options are: • Default (16bit) • signed 8bit (-TI1) • signed 16bit (-TI2) • signed 32bit (-TI4)
long	Selects the size of the long type. Options are: • Default (32bit) • signed 8bit (-TL1) • signed 16bit (-TL2) • signed 32bit (-TL4)
long long	Selects the size of the long long type. Options are:

Table continues on the next page...

Table 3-43. Tool Settings - HCS08 Compiler > Type Sizes (continued)

Option	Description
	• Default (32bit) • signed 8bit (-TLL1) • signed 16bit (-TLL2) • signed 32bit (-TLL4)
enum	Selects the size of the enum type. Options are: • Default (signed 16bit) • signed 8bit (-TE1sE) • signed 16bit (-TE2sE) • signed 32bit (-TE4sE) • unsigned 8bit (-TE1uE)
float	Selects the size of the float type. Options are: • Default (IEEE32) • IEEE32 • IEEE64
double	Selects the size of the double type. Options are: • Default (IEEE64) • IEEE32 • IEEE64
long double	Selects the size of the long double type. Options are: • Default (IEEE64) • IEEE32 • IEEE64
long long double	Selects the size of the long long double type. Options are: • Default (IEEE64) • IEEE32 • IEEE64

3.4.5.9 HCS08 Compiler > General

Use this panel to specify other flags for the HC(S)08 Compiler to use.

The following table lists and describes the **General** options for HC(S)08 compiler.

Table 3-44. Tool Settings - HC08 Compiler > General Options

Option	Description
Other flags	Specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.

3.4.5.10 HCS08 Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The table below lists and describes the **Optimization** options for HC(S)08 compiler.

Table 3-45. Tool Settings - HCS08 Compiler > Optimization Options

Option	Description
Disable optimization (-O0)	Disables all optimizations.
No integral promotion on characters (-Cni)	Enhances character operation code density by omitting integral promotion. This option enables behavior that is not ANSI-C compliant. Code generated with this option set does not conform to ANSI standards. Code compiled with this option is not portable. Using this option is not recommended in most cases.
Loop unrolling (i[number]) (-Cu)	Enables loop unrolling with the following restrictions: • Only simple for statements are unrolled, for example, for (i=0; i<10; i++) • Initialization and test of the loop counter must be done with a constant. • Only <, >, <=, >= are permitted in a condition. • Only ++ or -- are allowed for the loop variable increment or decrement. • The loop counter must be integral. • No change of the loop counter is allowed within the loop. • The loop counter must not be used on the left side of an assignment. • No address operator (&) is allowed on the loop counter within the loop. • Only small loops are unrolled: Loops with few statements within the loop. Loops with fewer than 16 increments or decrements of the loop counter. The bound may be changed with the optional argument = i<number>. The -Cu=i20 option unrolls loops with a maximum of 20 iterations.
Main Optimize Target: Optimize for	There are various points where the Compiler has to select between two possibilities: it can either generate fast, but large code, or small but slower code. The Compiler generally optimizes on code size. It often has to decide between a runtime routine or an expanded code. The programmer can decide whether to select between the slower and shorter or the faster and longer code sequence by setting a command line switch.

Table continues on the next page...

Table 3-45. Tool Settings - HCS08 Compiler > Optimization Options (continued)

Option	Description
	• The **Code Size (-Os)** option directs the Compiler to optimize the code for smaller code size. The Compiler trades faster-larger code for slower-smaller code. • The **Execution Time (-Ot)** option directs the Compiler to optimize the code for faster execution time. The Compiler replaces slower/smaller code with faster/ larger code. This option only affects some special code sequences. This option has to be set together with other optimization options (e.g., register optimization) to get best results.
Optimize dead assignments	Optimizes dead assignments. The Compiler removes assignments to unused local variables. There are three possible settings for this option: • **always (even if HLI present in function):** Always optimize dead assignments (even if HLI is present in current function). The Compiler does not consider inline assembler accesses. **Note:** This option is unsafe when inline assembler code contains accesses to local variables. • **yes, but never if HLI present in function** : No optimization occurs. This generates the best possible debug information, and produces larger and slower code. • **never** : Optimize dead assignments if HLI is not present in the current function.
Create sub-functions with common code	Performs the reverse of inlining. It detects common code parts in the generated code. The Compiler moves the common code to a different place and replaces all occurrences with a JSR to the moved code. At the end of the common code, the Compiler inserts an RTS instruction. The Compiler increases all SP uses by an address size. This optimization takes care of stack allocation, control flow, and of functions having arguments on the stack. Inline assembler code is never treated as common code. Options are: • **Default** • **Disable (-Onf)** • **Off (-Of)**
Dynamic options configuration for functions (-OdocF)	Allows the Compiler to select from a set of options to reach the smallest code size for every function. Without this feature, you must set fixed Compiler switches over the whole compilation unit. With this feature, the Compiler finds the best option combination from a user-defined set for every function.
Inlining (C[n] or OFF) (-Oi)	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. Using the -Oi=c0 option switches off inlining. Functions marked with the #pragma INLINE are still inlined. To disable inlining, use the -Oi=OFF option.

Table continues on the next page...

Table 3-45. Tool Settings - HCS08 Compiler > Optimization Options (continued)

Option	Description
Disable alias checking (-Ona)	Prevents the Compiler from redefining these variables, which allows you to reuse already-loaded variables or equivalent constants. Use this option only when you are sure no real writes of aliases to a variable memory location will occur.
Do generate copy down information for zero values (-OnCopyDown)	Using this option, the compiler does not generate a copy down for i. The initialization with zero optimization shown for the arr array only works in the HIWARE format. The ELF format requires initializing the whole array to zero.
Disable CONST variable by constant replacement (-OnCstVar)	Allows you to switch OFF the replacement of CONST variable by the constant value.
Disable code generation for NULL Pointer to Member check (-OnPMNC)	Before assigning a pointer to a member in C++, you must ensure that the pointer to the member is not NULL in order to generate correct and safe code. In embedded systems development, the difficulty becomes generating the denser code while avoiding overhead whenever possible (this NULL check code is a good example). This option enables you to switch off the code generation for the NULL check.
Large return value type	Compiler supports this option even though returning a 'large' return value may be not as efficient as using an additional pointer. The Compiler introduces an additional parameter for the return value if the return value cannot be passed in registers. Options are: • Default • Large return value pointer, always with temporary (-Rpt) • Large return value pointer and temporary elimination (-Rpe)
Optimize bitfields and volatile bitfields	Use this option to optimize bitfields and volatile bitfields. The compiler changes the access order or combines many accesses into one, even if the bitfields are declared as volatile.
Keep loop induction variables in registers	Limits the number of loop induction variables the Compiler keeps in registers. Specify any number down to zero (no loop induction variables). The compiler reads and writes loop induction variables within the loop (for example, loop counter), and attempts to keep the variables in registers to reduce execution time and code size. The Compiler takes the optimal number (code density) when this option is not specified. Specifying a high number of loop induction variables may increase code size, particularly for spill and merge code.
Disable optimize bitfields	Prevents the Compiler from combining sequences of bitfield assignments containing constants. This simplifies debugging and makes the code more readable.
Disable ICG level branch tail merging	Switches the ICG level branch tail merging off. This simplifies debugging and produces more readable code.
Disable any constant folding	Prevents the Compiler from folding constants over statement boundaries. All arithmetical operations are coded. This option must be set when using the library functions setjmp() and longjmp(), or the Compiler makes wrong assumptions.
Disable constant folding in the case of a new constant	This option prevents the Compiler from folding constants when the resulting constant is new. The option affects only

Table continues on the next page...

Table 3-45. Tool Settings - HCS08 Compiler > Optimization Options (continued)

Option	Description
	those processors where constants are difficult to load (e.g., RISC processors). On other processors this option makes no change.
Disable any low level common subexpression elimination	Prevents the Compiler from reusing common subexpressions, such as array indexes and array base addresses. The code size may increase. The low-level CSE does not have the alias problems of the frontend CSE and is therefore switched on by default. The two CSE optimizations do not cover the same cases. The low-level CSE has a finer granularity but does not handle all cases of the frontend CSE. Use this option only to generate more readable code for debugging.
Allocate local variables into registers	Allocates local variables (char or int) in registers. The number of local variables allocated in registers depends on the number of available registers. Use this option when using variables as loop counters or switch selectors or when the processor requires register operands for multiple operations (for example, RISC processors). Compiling with this option may increase your code size (spill and merge code). This optimization may increase code complexity when using High-Level Languages, making debugging more difficult.
Disable frame pointer optimization (-OnX)	Prevents the Compiler from converting stack pointer-relative accesses into X-relative accesses. The frame optimizer tries to convert all SP-relative accesses (local variables, spills) into shorter and faster X-relative accesses. In addition, the Compiler traces the value of H:X and removes useless TSX and AIX instructions. Using -OnX to switch the frame optimizer off facilitates debugging.

3.4.5.10.1 HCS08 Compiler > Optimization > Tree optimizer

The Compiler contains a special optimizer which optimizes the internal tree data structure. This tree data structure holds the semantic of the program and represents the parsed statements and expressions.

This option disables the tree optimizer. This may be useful for debugging and for forcing the Compiler to produce `straightforward' code.

Use this panel to configure the tree optimizer options for the HCS08 compiler.

The following table lists and describes the **Tree optimizer** options for HC(S)08 compiler.

Table 3-46. Tool Settings - HCS08 Compiler > Optimization > Tree optimizer

Option	Description
Disable all optimizations (-Ont)	Disable all the optimizations.
Disable bit neg optimization (-Ont=~)	Disable optimization of `~~i' into `i'.
Disable bit or optimization (-Ont=|)	Disable optimization of `i|0xffff' into `0xffff'.
Disable exor optimization (-Ont=^)	Disable optimization of `i^0' into `i'.
Disable if optimization (-Ont=w)	Disable optimization of `if (1) i = 0;' into `i = 0;'.
Disable do optimization (-Ont=v)	Disable optimization of `do ... while(0) into `...'.
Disable while optimization (-Ont=u)	Disable optimization of `while(1) ...;' into `...;'.
Disable for optimization (-Ont=t)	Disable optimization of `for(;;) ...' into `while(1) ...'.
Disable indirect optimization (-Ont=s)	Disable optimization of `*&i' into `i'.
Disable 16-32 relative optimization (-Ont=r)	Disable optimization of `L<=4' into 16-bit compares if 16-bit compares are better.
Disable 16-32 compare optimization (-Ont=q)	Reduction of long compares into int compares if int compares are better: (-Ont=q to disable it).
Disable cut optimization (-Ont=p)	Disable optimization of `(char)(long)i' into `(char)i'.
Disable cast optimization (-Ont=o)	Disable optimization of `(short)(int)L' into `(short)L' if short and int have the same size.
Disable right shift optimization (-Ont=n)	Optimization of shift optimizations (<<, -Ont=n to disable it)
Disable left shift optimization (-Ont=m)	Optimization of shift optimizations (>>, -Ont=m to disable it)
Disable label optimization (-Ont=l)	Disable optimization removal of labels if not used.
Disable transformations for inlining optimization (-Ont=j)	This optimization transforms the syntax tree into an equivalent form in which more inlining cases can be done. This option only has an effect when inlining is enabled.
Disable address optimization (-Ont=i)	Disable optimization of `&*p' into `p'.
Disable unary minus optimization (-Ont=h)	Disable optimization of `-(-i)' into `i'.
Disable compare size optimization (-Ont=g)	Disable optimization of compare size.
Disable condition optimization (-Ont=f)	Disable optimization of `(a==0)' into `(!a)'.
Disable const swap optimization (-Ont=e)	Disable optimization of `2*i' into `i*2'.
Disable binary operation optimization (-Ont=d)	Disable optimization of `us & ui' into `us & (unsigned short) ui'.
Disable compare optimization (-Ont=c)	Disable optimization of `if ((long)i)' into `if (i)'.
Disable constant folding optimization (-Ont=b)	Disable optimization of `3+7' into `10'.
Disable statement optimization (-Ont=a)	Disable optimization of last statement in function if result is not used.
Disable test optimization (-Ont=?)	Disable optimization of `i = (int)(cond ? L1:L2);' into `i = cond ? (int)L1:(int)L2;'.
Disable assign optimization (-Ont=9)	Disable optimization of `i=i;'.
Disable switch optimization (-Ont=8)	Disable optimization of empty switch statement.
Disable extend optimization (-Ont=7)	Disable optimization of `(long)(char)L' into `L'.
Disable or optimization (-Ont=1)	Disable optimization of `a || 0' into `a'.
Disable and optimization (-Ont=0)	Disable optimization of `a && 1' into `a'.

Table continues on the next page...

Table 3-46. Tool Settings - HCS08 Compiler > Optimization > Tree optimizer (continued)

Option	Description
Disable div optimization (-Ont=/)	Disable optimization of `a/1' into `a'.
Disable minus optimization (-Ont=-)	Disable optimization of `a-0' into `a'.
Disable plus optimization (-Ont=+)	Disable optimization of `a+0' into `a'.
Disable mul optimization (-Ont=*)	Disable optimization of `a*1' into `a'.
Disable bit and optimization (-Ont=)	Disable optimization of `a&0' into `0'.
Disable mod optimization (-Ont=%)	Disable optimization of `a%1' into `0'.

3.4.5.10.2 HCS08 Compiler > Optimization > Optimize Library Function

This option enables the compiler to optimize specific known library functions to reduce execution time. The Compiler frequently uses small functions such as strcpy(), strcmp(), and so forth. Use this panel to configure the optimize library function options for the HCS08 compiler.

The following table lists and describes Optimize Library Function options for HC(S)08 compiler.

Table 3-47. Tool Settings - HCS08 Compiler > Optimization > Optimize Library Function

Option	Description
Apply all optimizations (-OiLib)	This option applies all the optimizations.
shifts left of 1 (-OiLib=g)	This option replace shifts left of 1 by array lookup.
memcpy (-OiLib=f)	This option inline calls to the memcpy() function.
memset (-OiLib=e)	This option inline calls to the memset() function.
fabs/fabsf (-OiLib=d)	This option inline calls to the fabs() or fabsf() functions.
strlen (-OiLib=b)	This option inline calls to the strlen() function.

3.4.5.10.3 HCS08 Compiler > Optimization > Branch Optimizer

Use this panel to specify the branch optimizer options of the HCS08 compiler.

The following table lists and describes the **Branch Optimizer** options for HC(S)08 compiler.

Table 3-48. Tool Settings - HCS08 Compiler > Optimization > Branch Optimizer

Option	Description
Disable all optimizations (-OnB)	With this option, all low-level branch optimizations are disabled.
Disable tail branch optimization (-OnB=t)	Disable Branch tail optimization
Disable branch to RTS optimization (-OnB=r)	Disable Branch to RTS optimization
Disable long branch optimization (-OnB=l)	Disable long branch optimization
Disable dead code optimization (-OnB=d)	Disable dead code optimization
Disable Branch JSR to BSR optimization (-OnB=b)	Disable Branch JSR to BSR optimization
Disable short BRA optimization (-OnB=a)	Disable short BRA optimization

3.4.5.10.4 HCS08 Compiler > Optimization > Peephole Optimization

Use this panel to configure peephole optimization for the HC(S)08 Compiler.

The following table lists and describes the **Peephole Otimization** options for HC(S)08 compiler.

Table 3-49. Tool Settings - HCS08 Compiler > Optimization > Peephole Optimization

Option	Description
Disable all optimizations (-OnP)	If -OnP is specified, the Compiler disables the whole peephole optimizer.
Disable peephole load immediate to HX (HCS08 only) (-OnP=x)	Disable peephole load immediate to HX (HCS08 only).
Disable peephole simple inline assembler optimizations (-OnP=o)	Disable peephole simple inline assembler optimizations.
Disable peephole CMP #1 optimization (-OnP=n)	Disable peephole CMP #1 optimization.
Disable peephole JSR to JMP optimization (-OnP=m)	Disable peephole JSR to JMP optimization.
Disable peephole unnescessary transfers optimization (-OnP=l)	Disable peephole unnescessary transfers optimization.
Disable peephole unnescessary tests optimization (-OnP=k)	Disable peephole unnescessary tests optimization.
Disable peephole unused compares optimization (-OnP=j)	Disable peephole unused compares optimization.
Disable peephole unused stores optimization (-OnP=i)	Disable peephole unused stores optimization.
Disable peephole unused loads optimization (-OnP=h)	Disable peephole unused loads optimization.
Disable peephole RTS RTS optimization (-OnP=g)	Disable peephole RTS RTS optimization.
Disable peephole PSH PUL optimization (-OnP=f)	Disable peephole PSH PUL optimization.
Disable peephole combine bit set/clr optimization (-OnP=e)	Disable peephole combine bit set/clr optimization.
Disable peephole combine bit operations optimization (-OnP=d)	Disable peephole combine bit operations optimization.

Table continues on the next page...

Table 3-49. Tool Settings - HCS08 Compiler > Optimization > Peephole Optimization (continued)

Option	Description
Disable peephole PSH/PUL instead AIS optimization (-OnP=c)	Disable peephole PSH/PUL instead AIS optimization.
Disable peephole handle constant argument optimization (-OnP=b)	Disable peephole handle constant argument optimization.
Disable peephole combine AI(S|X) optimization (-OnP=a)	Disable peephole combine AI(S|X) optimization.

3.4.6 HCS08 Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler.

The following table lists and describes the assembler options for HCS08.

Table 3-50. Tool Settings - Assembler Options

Option	Description
Command	Shows the location of the assembler executable file. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS}- Objn${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.
Command line pattern	

3.4.6.1 HCS08 Assembler > Output

Use this panel to control how the assembler generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the output options for HCS08 Assembler.

Table 3-51. Tool Settings - HCS08 Assembler > Output Options

Option	Description
Object File Format (-F)	Defines the object file format.
Show label statistics (-Ll)	Enables the Compiler to append statistical information about the compilation session to the specified file. The information includes Compiler options, code size (in bytes), stack usage (in bytes) and compilation time (in seconds) for each procedure of the compiled file. The Compiler appends the information to the specified filename (or the file make.txt, if no argument given). Set the TEXTPATH: Text File Path environment variable to store the file into the path specified by the environment variable. Otherwise the Compiler stores the file in the current directory.
Generate listing file (for example, %(TEXTPATH)/%n.lst) (-L)	Specifies the name, %n, of the assembly listing file. The file is placed in the directory specified by %TEXTPATH. If this option is left blank, no listing file is output.
Address size in the listing file (integer) (-Lasms)	Specifies the size of the addresses displayed in the listing. Options are: • 1 to display addresses as xx • 2 to display addresses as xxxx • 3 to display addresses as xxxxxx • 4 to display addresses asf xxxxxxxx
Do not print macro call in listing file (-Lc)	Specifies whether macro calls encountered in the source code are expanded and appear in the listing file.
Do not print macro definition in listing file (-Ld)	Instructs the Assembler to generate a listing file but not including any macro definitions. The listing file contains macro invocation and expansion lines as well as expanded include files.
Do not print macro expansion in listing file (-Le)	Switches on the generation of the listing file, but macro expansions are not present in the listing file. The listing file contains macro definition and invocation lines as well as expanded include files.
Do not print included files in listing file (-Li)	Switches on the generation of the listing file, but include files are not expanded in the listing file. The listing file contains macro definition, invocation, and expansion lines.

3.4.6.1.1 HCS08 Assembler > Output > Configure listing file

Use this panel to specify the general assembler behavior.

The following table lists and describes the configure listing file options for HCS08.

Table 3-52. Tool Settings - Assembler > Output > Configure listing file Options

Option	Description
Disable all (-Lasmc)	Disables printing of all the columns in the listing file

Table continues on the next page...

Table 3-52. Tool Settings - Assembler > Output > Configure listing file Options (continued)

Option	Description
Do not write the source line (-Lasmc=s)	Do not print source column in the listing file
Do not write the relative line (-Lasmc=r)	Do not print relative column (Rel.) in the listing file
Do not write the macro mark (-Lasmc=m)	Do not print macro mark column in the listing file
Do not write the address (-Lasmc=l)	Do not print address column (Loc) in the listing file
Do not write the location kind (-Lasmc=k)	Do not print the location type column in the listing file
Do not write the include mark column (-Lasmc=i)	Do not print the include mark column in the listing file
Do not write the object code (-Lasmc=c)	Do not print the object code in the listing file
Do not write the absolute line (-Lasmc=a)	Do not print the absolute column (Abs.) in the listing file

3.4.6.2 HCS08 Assembler > Input

Use this panel to specify file search paths and any additional include files the HCS08 Assembler should use. You can specify multiple search paths and the order in which you want to perform the search.

The following table lists and describes the input options for HCS08 Assembler.

Table 3-53. Tool Settings - HCS08 Assembler > Input Options

Option	Description
Include File Search Paths (-I)	Lists the included file search paths.
Case sensitivity or label name (-Ci)	Check to make the label names case sensitive.
Define label (Use spaces to separate labels) (-D)	Define labels that have to be included in the RS08 assembler input.
Support for structured types (-Struct)	Check to include the support for structured types.

The following table lists and describes the toolbar buttons that help work with the file search paths.

Table 3-54. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the file search path.
✖	Delete - Click to delete the selected file search path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.

Table continues on the next page...

Table 3-54. Search Paths Toolbar Buttons (continued)

Button	Description
⇧	Move up - Click to move the selected file search path one position higher in the list.
⇩	Move down - Click to move the selected file search path one position lower in the list.

3.4.6.3 HCS08 Assembler > Language

Use this panel to specify code- and symbol-generation options for the HCS08 Assembler.

The following table lists and describes the language options for HCS08 Assembler.

Table 3-55. Tool Settings - HCS08 Assembler > Language Options

Option	Description
Angle brackets for macro arguments grouping (-CMacAngBrack)	Controls whether the < > syntax for macro invocation argument grouping is available. When it is disabled, the Assembler does not recognize the special meaning for < in the macro invocation context. There are cases where the angle brackets are ambiguous. In new code, use the [? ?] syntax instead. Options are: • Allow • Disallow
Square braces for macro arguments grouping (-CMacBrackets)	Controls the availability of the [? ?] syntax for macro invocation argument grouping. When it is disabled, the Assembler does not recognize the special meaning for [? in the macro invocation context. Options are: • Allow • Disallow
Maximum MacroNest nesting (-MacroNest)	Controls how deep macros calls can be nested. Its main purpose is to avoid endless recursive macro invocations.

3.4.6.3.1 HCS08 Assembler > Language > Compatibility modes

Use this panel to specify language compatibility modes for the HCS08 Assembler.

The following table lists and describes the compatibility mode options for HCS08 Assembler.

Table 3-56. Tool Settings - HCS08 Assembler > Language > Compatibility mode Options

Option	Description
Select all (-Compat)	Check to enable all compatibility mode options.
Symbol prefixes (-Compat=s)	With this suboption, the Assembler accepts "pgz:" and "byte:" prefixed for symbols in XDEFs and XREFs. They correspond to XREF.B or XDEF.B with the same symbols without the prefix.
Ignore FF character at line start Symbol prefixes (-Compat=f)	With this suboption, an otherwise improper character recognized from feed character is ignored.
Alternate comment rules (-Compat=c)	With this suboption, comments implicitly start when a space is present after the argument list. A special character is not necessary. Be careful with spaces when this option is given because part of the intended arguments may be taken as a comment. However, to avoid accidental comments, the Assembler does issue a warning if such a comment does not start with a "*" or a ";".
Support FOR directive (-Compat=b)	With this suboption, the Assembler supports a FOR - Repeat assembly block assembly directive to generate repeated patterns more easily without having to use recursive macros.
Add some additional directives (-Compat=a)	With this suboption, some additional directives are added for enhanced compatibility. The Assembler actually supports a SECT directive as an alias of the usual SECTION - Declare Relocatable Section assembly directive. The SECT directive takes the section name as its first argument.
Operator != means equal (-Compat==)	The Assembler takes the default value of the != operator as not equal, as it is in the C language. For compatibility, this behavior can be changed to equal with this option. Because of the risks involved with this option for existing code, a message is issued for every != which is treated as equal.
Support $ character in symbol (-Compat=)	With this suboption, the Assembler supports to start identifiers with a $ sign.
Support additional ! symbols (-Compat=!)	The following additional operators are defined when this option is used: • !^: exponentiation • !m: modulo • !@: signed greater or equal • !g: signed greater • !%: signed less or equal • !t: signed less than • !$: unsigned greater or equal • !S: unsigned greater • !&: unsigned less or equal • !l: unsigned less • !n: one complement • !w: low operator • !h: high operator **Note:** The default values for the following ! operators are defined: • !.: binary AND

Table 3-56. Tool Settings - HCS08 Assembler > Language > Compatibility mode Options

Option	Description
	• !x: exclusive OR • !+: binary OR

3.4.6.4 HCS08 Assembler > Host

Use this panel to specify the host settings of the HCS08.

The following table lists and describes the memory model options for HCS08.

Table 3-57. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.4.6.5 HCS08 Assembler > Code Generation

Use this panel to specify the code generation assembler behavior.

The following table lists and describes the code generation assembler options for HCS08.

Table 3-58. Tool Settings - Assembler > Code Generation Options

Option	Description
Associate debug information to assembly source file (-AsmDbg)	Passes the assembly source file name information to DWARF sections. When the output .abs file is debugged, the actual assembly source file is displayed instead of intermediary <filename>.dbg file.

3.4.6.6 HCS08 Assembler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-59. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Assembler uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Assembler with additional arguments (for example, files and Assembler options). If you start the Assembler with arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the Assembler compiles the files in a batch mode. No Assembler window is visible and the Assembler terminates after job completion.
Message Format for batch Mode (e.g. %"%f%e%"(%l): %K %d: %m\n)(-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`)

Table continues on the next page...

Table 3-59. Tool Settings - Messages Options (continued)

Option	Description
	• %n: filename (example, mysourcefile) • %e: Extension (example, .cpph) • %N: File (8 chars) (example, mysource) • %E: Extension (3 chars) (example, .cpp) • %l: Line (example, 3) • %c: Column (example, 47) • %o: Pos (example, 1234) • %K: Uppercase kind (example, ERROR) • %k: Lowercase kind (example, error) • %d: Number (example, C1815) • %m: Message (example, text) • %%: Percent (example, %) • \n: New line • %": A " if the filename, the path, or the extension contains a space • %': A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Assembler creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.4.6.6.1 HCS08 Assembler > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-60. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.4.6.7 HCS08 Assembler > General

Use this panel to specify the general assembler behavior.

The following table lists and describes the general assembler options for HCS08.

Table 3-61. Tool Settings - Assembler > General Options

Option	Description
MMU Support (-MMU)	Check to inform the compiler that CALL and RTC instructions are available, enabling code banking, and that the current architecture has extended data access capabilities, enabling support for `__linear` data types. This option can be used only when `-Cs08` is enabled.
MCUasm compatibility (-MCUasm)	Check to activate the compatibility mode with the MCUasm Assembler.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.

3.4.7 HCS08 Preprocessor

Use this panel to specify the preprocessor settings of the HCS08.

The following table lists and describes the preprocessor options for HCS08.

Table 3-62. Tool Settings - Preprocessor Options

Option	Description
Command	Shows the location of the preprocessor executable file. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} -Lp ${FLAGS} ${INPUTS}`.
Command line pattern	

3.4.7.1 HCS08 Preprocessor > Preprocessor Settings

Use this panel to specify the preprocessor settings of HCS08.

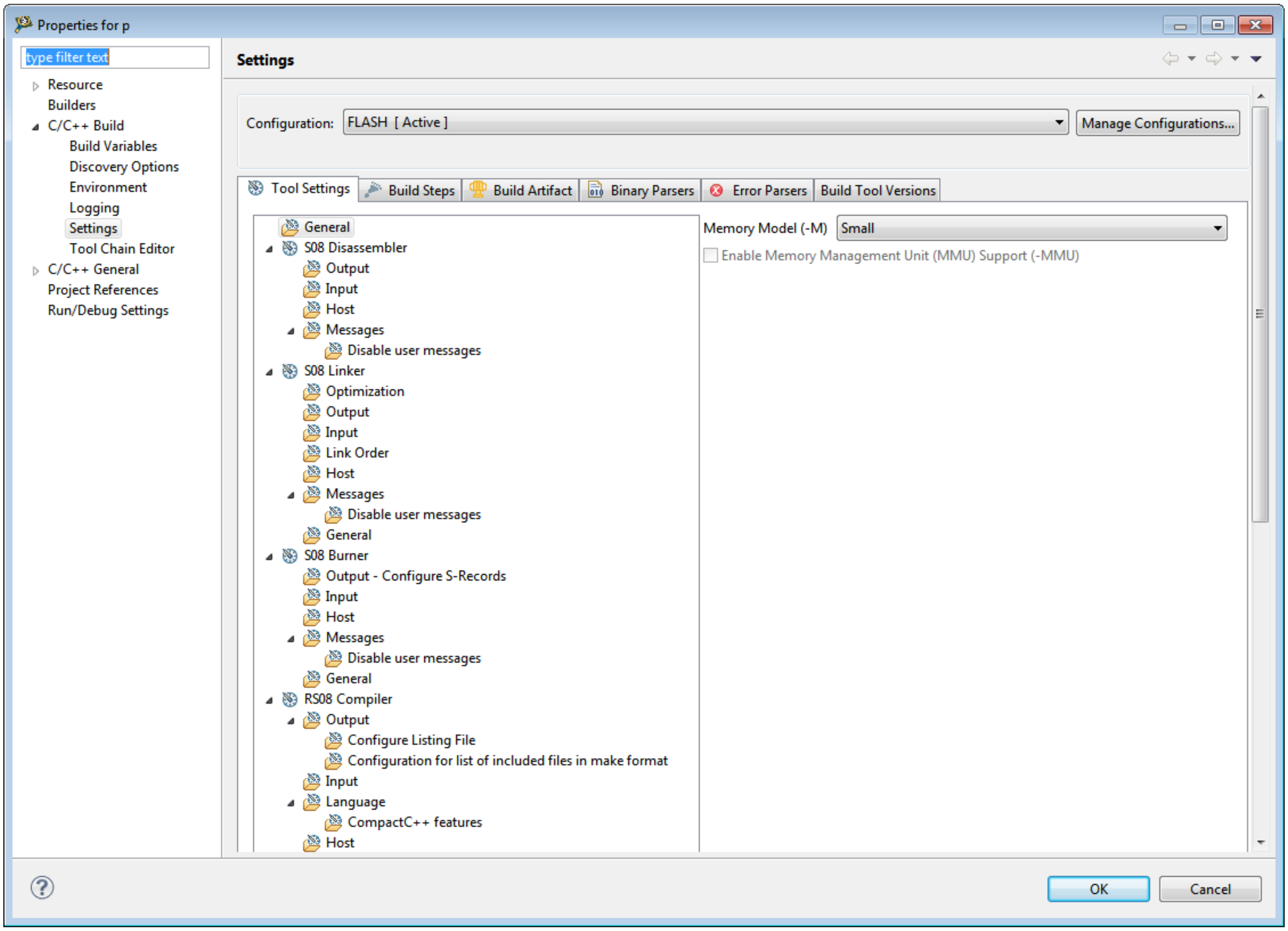
The following table lists and describes the preprocessor settings options for HCS08.

Table 3-63. Tool Settings - Preprocessor > Preprocessor Settings Options

Option	Description
Turn on all preprocessor configuration	Use this option to enable the default preprocessor configuration.
Emit whitespaces (-LpCfg=s)	Use this option to reconstruct spaces.
Handle single quote (`) as normal token (-LpCfg=q)	Use this option to handle single quote (`) as normal token.
Do not concatenate strings (-LpCfg=n)	Use this option to avoid string concatenation.
Emit #line directive (-LpCfg=l)	Use this option to emit #line directives in preprocessor output.
Do not emit file names (-LpCfg=m)	Do not emit file names.
Emit file names with path (-LpCfg=f)	Use this option to emit file names with path.
Emit empty lines (-LpCfg=e)	Use this option to emit empty lines.
Do not emit line comments (-LpCfg=c)	Do not emit line comments
Stop after preprocessor (-LpX)	Without this option, the compiler always translates the preprocessor output as C code. To do only preprocessing, use this option together with the -Lp option. No object file is generated.

3.5 Build Properties for RS08

The **Properties for** <*project*> window shows the corresponding build properties for an RS08 project.



Configuration: FLASH [Active]

Memory Model (-M) Small

☐ Enable Memory Management Unit (MMU) Support (-MMU)

Figure 3-12. Build Properties - RS08

The following table lists the build properties specific to developing software for HCS08.

The properties that you specify in these panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> window.

Table 3-64. Build Properties for RS08

Build Tool	Build Properties Panels
General	General
S08 Disassembler	S08 Disassembler > Output
	S08 Disassembler > Input
	S08 Disassembler > Host

Table continues on the next page...

Table 3-64. Build Properties for RS08 (continued)

Build Tool	Build Properties Panels
	S08 Disassembler > Messages
	S08 Disassembler > Messages > Disable user messages
S08 Linker	S08 Linker > Optimization
	S08 Linker > Output
	S08 Linker > Input
	S08 Linker > Host
	S08 Linker > Messages
	S08 Linker > Messages > Disable user messages
	S08 Linker > General
S08 Burner	S08 Burner > Output > Configure S-Record
	S08 Burner > Input
	S08 Burner > Host
	S08 Burner > Messages
	S08 Linker > Messages > Disable user messages
	S08 Burner > General
RS08 Compiler	RS08 Compiler > Output
	RS08 Compiler > Output > Configure Listing File
	RS08 Compiler > Output > Configuration for list of included files in make format
	RS08 Compiler > Input
	RS08 Compiler > Language
	RS08 Compiler > Language > CompactC++ features
	RS08 Compiler > Host
	RS08 Compiler > Code Generation
	RS08 Compiler > Messages
	RS08 Compiler > Messages > Disable user messages
	RS08 Compiler > Preprocessor
	RS08 Compiler > Type Sizes
	RS08 Compiler > General
	RS08 Compiler > Optimization
	RS08 Compiler > Optimization > Mid level optimizations
	RS08 Compiler > Optimization > Mid level branch optimizations
	RS08 Compiler > Optimization > Tree optimizer
	RS08 Compiler > Optimization > Optimize Library Function
RS08 Assembler	RS08 Assembler > Output
	RS08 Assembler > Output > Configure Listing File
	RS08 Assembler > Input
	RS08 Assembler > Language
	RS08 Assembler > Language > Compatibility modes

Table continues on the next page...

Table 3-64. Build Properties for RS08 (continued)

Build Tool	Build Properties Panels
	RS08 Assembler > Host
	RS08 Assembler > Code Generation
	RS08 Assembler > Messages
	RS08 Assembler > Messages > Disable user messages
	RS08 Assembler > General
RS08 Preprocessor	RS08 Preprocessor > Preprocessor Settings

3.5.1 General

Use this panel to specify the memory model that the architecture uses. The build tools (compiler, linker, and assembler) use the properties that you specify.

The following table lists and describes the memory model options for RS08.

Table 3-65. Tool Settings - General

Option	Description
Memory Model (-M)	Specify the memory model for the build tools: • **Tiny** - Assumes that data pointers have 8-bit addresses unless explicitly specified with the keyword __far • **Small** - Default memory model; assumes that all functions and pointers have 16 bit addresses and requires code and data to be located in 64 kilobytes address space • **Banked** - Lets you place program code into atmost 256 pages of 16 kilobytes each, but does not affect data allocation
Enable Memory Management Unit (MMU) Support (-MMU)	Check to inform the compiler that `CALL` and `RTC` instructions are available, enabling code banking, and that the current architecture has extended data access capabilities, enabling support for `__linear` data types. This option can be used only when `-Cs08` is enabled.

3.5.2 S08 Disassembler

Use this panel to specify the command, options, and expert settings for RS08 Disassembler.

The following table lists and describes the Disassembler options.

Table 3-66. Tool Settings - Disassembler Options

Option	Description
Command	Shows the location of the disassembler executable file; default is `${HC08Tools}/decoder`. You can specify additional command line options for the disassembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the disassembler will be called with.
Expert Settings	
Command line pattern	Shows the command line patterns; default is `${COMMAND} ${FLAGS} -O${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`

3.5.2.1 S08 Disassembler > Output

Use this panel to control how the disassembler generates the output file.

The following table lists and describes the output options for RS08 disassembler.

Table 3-67. Tool Settings - Disassembler > Output Options

Option	Description
Print full listing	Prints a listing with the header information of the object file.
Write disassembly listing with source code	Check to enable the decoder write the source code within the disassembly listing, when decoding Freescale object files. ELF object files are not affected by this option. This option setting is default for the Freescale object files as input.
Decode DWARF section	Check to write the DWARF section information in the listing file. Decoding from the DWARF section inserts this information in the listing file.
Configure which parts of DWARF in formation to decode	Check to configure which parts of DWARF information to decode.
Decode ELF sections	Check to ensure that the ELF section information is also written to the listing file. Decoding from the ELF section inserts information in the listing file.
Dump ELF sections	Check to generate a HEX dump of all ELF sections. Decode ELF sections and Dump ELF sections both refer to the same decoder option (-E).
Dump ELF sections in LST file	Check to generate a HEX dump of all ELF sections in a LST file. The related command -E produces the same information, but in a more readable form.
Produce inline assembly file	Check to ensure that the output listing is an inline assembly file without additional information, but in C comments. Only for Freescale object files.

Table continues on the next page...

Table 3-67. Tool Settings - Disassembler > Output Options (continued)

Option	Description
No symbols in disassembled listing	Check to prevent symbols from printing in the disassembled listing.
Shows the cycle count for each instruction	Check to ensure that each instruction line contains the cycle count in '[',']' braces. The cycle count is written before the mnemonics of the instruction. Note that the cycle count display is not supported for all architectures.
Write disassembly listing only	Check to ensure that the Decoder writes the pure dissasembly only within the listing, without any source or comments. For Freescale object files only.
Write disassembly listing with source and all comments	Check to ensure the Decoder also includes the original source and comments in the disassembly listing. For Freescale objects only..

3.5.2.2 S08 Disassembler > Input

Use this panel to control how the disassembler generates the input file.

The following table lists and describes the input options for RS08 disassembler.

Table 3-68. Tool Settings - Disassembler > Input Options

Option	Description
Object File Format	Defines the format of the input object files.
Set processor	Specifies which processor of the input object file generated for. For object files, libraries and applications, the processor is usually detected automatically. For S-Record and Intel Hex files, however, the decoder cannot determine which CPU the code is for, and therefore the processor must be specified with this option to get a disassembly output. Without this option, only the structure of a S-Record file is decoded. The following values are supported: HC08, HC08:HCS08, HC11, HC12, HC12:CPU12, HC12:HCS12, HC12:HCS12X, HC16, M68k, MCORE, PPC, RS08, 8500, 8300, 8051 and XA.

3.5.2.3 S08 Disassembler > Host

Use this panel to specify the host settings of the RS08.

The following table lists and describes the memory model options for RS08.

Table 3-69. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.5.2.4 S08 Disassembler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-70. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.

Table continues on the next page...

Table 3-70. Tool Settings - Messages Options (continued)

Option	Description
Create err.log Error file	Using this option, the disassembler uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the disassembler with additional arguments (for example, files and disassembler options). If you start the disassembler with arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the disassembler compiles the files in a batch mode. No disassembler window is visible and the disassembler terminates after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the disassembler creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.

Table continues on the next page...

Table 3-70. Tool Settings - Messages Options (continued)

Option	Description
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.5.2.4.1 S08 Disassembler > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table table lists and describes the message options.

Table 3-71. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.5.3 S08 Linker

Use this panel to specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the general, libraries, and search path settings.

The following table lists and describes the linker options for RS08.

Table 3-72. Tool Settings - Linker Options

Option	Description
Command	Shows the location of the linker executable file. Default value is `"${HC08Tools}/linker.exe"`. You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} -add( ${INPUTS} )`

3.5.3.1 S08 Linker > Optimization

Use this panel to control linker optimizations. The linker's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the linker optimization options for HCS08 **.**

Table 3-73. Tool Settings - Linker > Optimization Options

Option	Description
Allocation over segment boundaries (-Alloc)	The linker supports to allocate objects from one ELF section into different segments. The allocation strategy controls where space for the next object is allocated as soon as the first segment is full. In the AllocNext strategy, the linker always takes the next segment as soon as the current segment is full. Holes generated during this process are not used later. With this strategy, the allocation order corresponds to the definition order in the object files. Objects defined first in a source file are allocated before later defined objects. In the AllocFirst strategy, the linker checks for every object, if there is a previously only partially used segment, into which the current object does fit. This strategy does not maintain the definition order. In the AllocChange strategy, the linker checks as soon as a object does no longer fit into the current segment, if there is a previously only partially used segment,

Table continues on the next page...

Table 3-73. Tool Settings - Linker > Optimization Options (continued)

Option	Description
	into which the current object does fit. This strategy does not maintain the definition order, but it does however use fewer different ranges than the AllocFirst case.
Allocate non referenced overlap variables (-CAllocUnusedOverlap)	When Smart Linking is switched off, defined but unreferenced overlapped variables are not allocated by default. Such variables do not belong to a specific function, therefore they cannot be allocated overlapped with other variables. This option only changes the behavior of variables in the special _OVERLAP segment. This segment is used only to allocate parameters and local variables for processors which do not have a stack. Not allocating an unreferenced overlap variable is similar to not allocating a variable on the stack for other processors. If you use this stack analogy, then allocating such variables this way corresponds to allocating unreferenced stack variables in global memory. This option allows allocation of all defined objects. Using this option is not recommended.
Enable automatic const placement (-ConstDist)	With this option the linker constant optimizer is enabled. Instead of performing usual linking actions, the linker generates a data distribution file which contains optimized distribution for constant objects.
Specify constant distribution segment name (-ConstDistSeg)	When this option is enabled, it's possible to specify the name of the constant distribution segment.
Allcoate non specified const segments in RAM (-CRam)	This option allocates constant data segments not explicitly allocated in a READ_ONLY segment in the default READ_WRITE segment. This was the default for old versions of the linker, so this option provides a compatible behavior with old linker versions.
Enable automatic data placement (-DataDist)	With this option the linker data optimizer is enabled. Instead of performing usual linking actions, the linker generates a data distribution file which contains optimized distribution.
Specify data distribution file name (-DataDistFile)	When this option is enabled, it's possible to specify the name of the data distribution file. There, all distributed data and how the compiler has to reallocate them are listed.
Generate data optimizer information file (-DataDistInfo)	When this option is enabled, the data optimizer generates a data distribution information file giving information on object to segment mapping
Specify data distribution segment name (-DataDistSeg)	When this option is enabled, it's possible to specify the name of the data distribution segment.
Enable distribution optimization (-Dist)	This option enables the linker optimizer. Instead of a link, the linker generates a distribution file which contains an optimized distribution.
Specify distribution file name (-DistFile)	Enable this option to specify the name of the distribution file. The distribution file lists all distributed functions and specifies how the compiler reallocates them.
Generate optimizer information file (-DistInfo)	Using this option, the optimizer generates a distribution information file containing a list of all sections and their functions. Available function information includes the old size, optimized size, and new calling convention.

Table continues on the next page...

Table 3-73. Tool Settings - Linker > Optimization Options (continued)

Option	Description
Choose optimizing method (-DistOpti)	Enable this option to choose the optimizing method. With the FillBanks argument the linker minimizes the free space in every bank. FillBanks is most effective for functions using the near calling convention. Use the CodeSize argument to minimize code when free space within the banks is no concern.
Specify distribution segment name (-DistSeg)	Use this option to specify the name of the distribution segment.
Specify library file name (-LibFile)	When this option is enabled,linker generates file<filename> which has information about the current libraries and also about the files with which they should be replaced with.
Enable library option file generation (_LibOptions)	Enables library information generation. When this option is enabled,linker generates file (default libFile.txt) which has information about the current library and the startup file and also about the files with which they should be replaced with.
Specify data optimizer options file name (-OptioneFile)	Specifies the name of the file that contains the set of linker-generated compiler options. When this option is enabled, linker places the second step compiler options in the specified file<filename>.
Enable option file generation (-Options)	Enables compiler option generation. The generated options will be used for second step compilation. Linker generates a text file containing a compiler option for the second step (one of the following: -ConstQualiNear, -NonConstQualiNear, -Mb). The content of the file is appended to the compiler options for the second compilation step.
Specify library file name (-P2LibFileName)	Specifies the name of the library information file. When this option is enabled in second link step,linker reads file<filename> which has information about the libraries.
Enable option to read libFile.txt in P2 (-ReadLibFile)	Instructs the linker to read in the library information file that it generated in step one. This option is passed in second link step. It tells the linker to read library information file(default libFile.txt).
Emit StartUp information to library info file (-StartUpInfo)	The information about the current startup file and the replacement startup file will be added to the library file(default libFile.txt) and used during the second compile-link step.
Overlap constants in ROM (-COCC)	Defines the default if constants and code should be optimized; commands `DO_OVERLAP_CONSTS` and `DO_NOT_OVERLAP_CONSTS` take precedence over the option.
Optimize copy down (-OCopy)	Changes the copy down structure to use few spaces. The optimization does assume that the application does perform both the zero out and the copy down step of the global initialization. If a value is set to zero by the zero out, then zero values are removed from the copy down information. The resulting initialization is not changed by this optimization if the default startup code is used.

3.5.3.2 S08 Linker > Output

Use this panel to control how the linker formats the listing file, as well as error and warning messages.

The following table lists and describes the linker output options for HCS08.

Table 3-74. Tool Settings - Linker > Output Options

Option	Description
Link as ROM library (-AsROMlib)	Check to link the application as a ROM library. This option has the same effect as specifying AS ROM_LIB in the linker parameter file.
Generate S_record file (-B)	Check to specify that in addition to an absolute file, also an srecord file should be generated. The name of the srecord file is the same as the name of the abs file, except that the extension "SX" is used. The default.env variable "SRECORD" may specify an alternative extension.
Check if objects overlap in the absolute file (even if different address spaces) (-CheckAcrossAddrSpace)	Check to instruct the linker to check if objects overlap, taking into account their address space.
Define the default value of the EPAGE register (-DefaultEpage)	Defines the reset value for the EEPROM Page Index Register (EPAGE). The value is specific to the actual S12(X) derivative.
Define the default value of the PPAGE register (-DefaultPpage)	Defines the reset value for the Program Page Index Register (PPAGE). The value is specific to the actual S12(X) derivative.
Define the default value of the RPAGE register (-DefaultRpage)	Defines the reset value for the RAM Page Index Register (RPAGE). The value is specific to the actual S12(X) derivative.
Generate map file (-M)	Check to scan source files for dependencies and emit a Makefile, without generating object code.
Never check section qualifier compatibility (-NoSectCompat)	For some target CPU's, when placing a section in a segment the linker checks if the qualifiers of the section are compatible with the ones of the segment (for instance when placing .text into RAM may result in a linker error).This option disables the check.
Strip symbolic information (-S)	Check to disable the generation of DWARF sections in the absolute file to save memory space.
Generate fixups in abs file (-SFixups)	Check to ensure compatibility with previous linker versions. Usually, absolute files do not contain any fixups because all fixups are evaluated at link time. But with fixups, the decoder might symbolically decode the content in absolute files. Some debuggers do not load absolute files which contain fixups because they assume that these fixups are not yet evaluated. But the fixups inserted with this option are actually already handled by this linker.
Enable Stack Consumption Computation (-StackConsumption)	The linker computes maximum stack effect for given application when the option is enabled and places the result in the output .map file.

Table continues on the next page...

Table 3-74. Tool Settings - Linker > Output Options (continued)

Option	Description
Specify statistic file (e.g. statistic.txt) (-StatF)	Specify the name of the linker statistic file. The statistic file reports each allocated object and its attributes. Every attribute is separated by a tab character, so it can be easily imported into a spreadsheet/database program for further processing.

3.5.3.3 S08 Linker > Input

Use this panel to specify the parameter file path, startup function, object file search paths, and any additional libraries that the C/C++ Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The IDE first looks for an include file in the current directory, or the directory that you specify in the `INCLUDE` directive. If the IDE does not find the file, it continues searching the paths shown in this panel. The IDE keeps searching paths until it finds the `#include` file or finishes searching the last path at the bottom of the Include File Search Paths list. The IDE appends to each path the string that you specify in the `INCLUDE` directive.

NOTE

> The IDE displays an error message if a header file is in a different directory from the referencing source file. Sometimes, the IDE also displays an error message if a header file is in the same directory as the referencing source file.

For example, if you see the message `Could not open source file myfile.h`, you must add the path for `myfile.h` to this panel.

The following table lists and describes the linker input options for HCS08.

Table 3-75. Tool Settings - Linker > Input Options

Option	Description
Parameter File	Shows the path of the parameter file. Default value is $ `{ProjDirPath}/Project_Settings/Linker_Files/Project.prm`.
Specify startup function (-E)	Defines the application entry point.
Search paths (-L)	Shows the list of all search paths; the ELF part of the linker searches object files first in all paths and then the usual environment variables are considered.

Table continues on the next page...

Table 3-75. Tool Settings - Linker > Input Options (continued)

Option	Description
Libraries	Lists paths to additional libraries that the C/C++ linker uses. Default value is `"${MCUToolsBaseDir}/lib/hc08c/lib/ansiis.lib"`
Link case insensitive	With this option, the linker ignores object name capitalization. This option supports case-insensitive linking of assembly modules. Since all identifiers are linked case insensitive, this also affects C or C++ modules. This option only affects the comparison of names of linked objects. Section names or the parsing of the link parameter file are unaffected. They remain case sensitive.
Object File Format	Defines the object file format.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object file search paths.

Table 3-76. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the Add directory path dialog box.

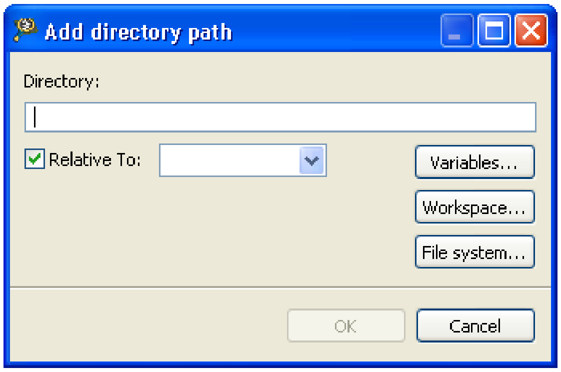
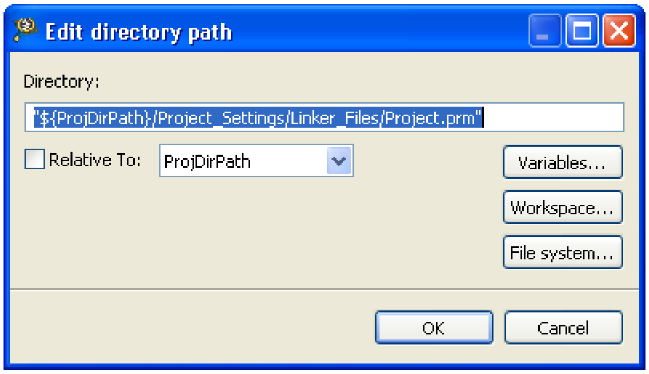


Figure 3-13. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



Directory: "${ProjDirPath}/Project_Settings/Linker_Files/Project.prm"

☐ Relative To: ProjDirPath

Figure 3-14. Edit directory path Dialog Box

The buttons in the **Add directory path and Edit directory path** dialog boxes help work with the object file search paths.

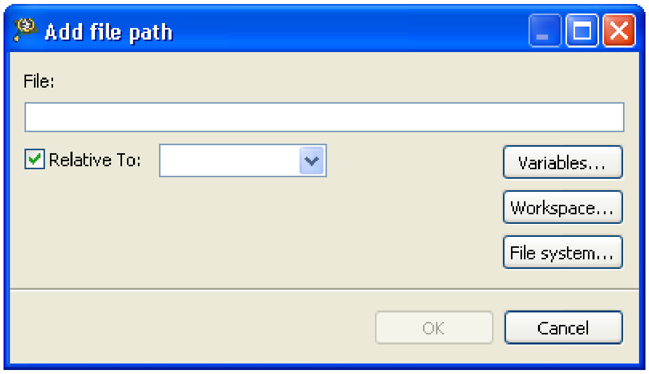
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the object file search path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object files.

Table 3-77. Libraries Toolbar Buttons

Button	Description
	Add - Click to open the **Add file path** dialog box and specify location of the library you want to add.
	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
	Edit - Click to open the **Edit file path** dialog box and update the selected path.
	Move up - Click to move the selected path one position higher in the list.
	Move down - Click to move the selected path one position lower in the list.

The following figure shows the Add file path dialog box.



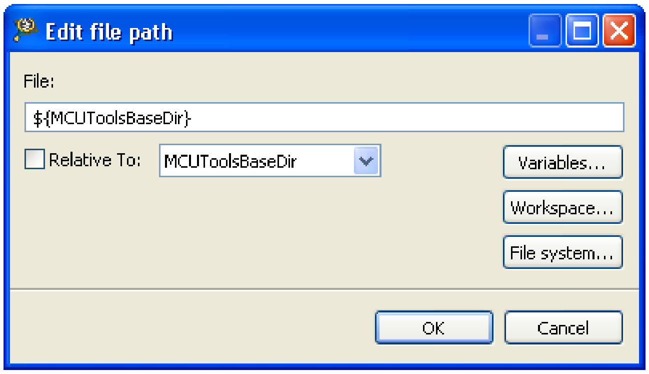
Add file path

File:

☑ Relative To:

Figure 3-15. Tool Settings - Linker > Libraries - Add file path Dialog Box

The following figure shows the Edit file path dialog box.



Edit file path

File: ${MCUToolsBaseDir}

□ Relative To: MCUToolsBaseDir

Figure 3-16. Tool Settings - Linker > Libraries - Edit file path Dialog Box

The buttons in the **Add file path and Edit file path** dialog boxes help work with the file paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **File Selection** dialog box and specify the file path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Open** dialog box and specify the file path. The resulting absolute path appears in the appropriate list.

3.5.3.4 S08 Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-78. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.5.3.5 S08 Linker > Host

Use this panel to specify the host settings of the RS08.

The following table lists and describes the memory model options for RS08.

Table 3-79. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.5.3.6 S08 Linker > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-80. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Linker uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Linker with additional arguments (for example, files and Linker). If you start the Linker arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the Linker the files in a batch mode. No Linker is visible and the Linker after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line

Table continues on the next page...

Table 3-80. Tool Settings - Messages Options (continued)

Option	Description
	• `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Linker creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.5.3.6.1 S08 Linker > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-81. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.

Table continues on the next page...

Table 3-81. Tool Settings - Disable user messages Options (continued)

Option	Description
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.5.3.7 S08 Linker > General

Use this panel to specify the general linker behavior.

The following table lists and describes the **general linker options for HCS08.**

Table 3-82. Tool Settings - Linker > General Options

Option	Description
Other flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI. Default value is: `-WmsgSd1100 -WmsgSd1912`

3.5.4 S08 Burner

Use the Burner for HCS08 Preference Panel to map *.bbl (batch burner language) files to the Burner Plug-In. When the project folder contains a *.bbl file, *.bbl file processing during the post-link phase uses the settings in the Burner preference panel.

The following table lists and describes the burner options for RS08.

Table 3-83. Tool Settings - Burner Options

Option	Description
Command	Shows the location of the burner executable file. Default value is: `"${HC08Tools}/burner"`. You can specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the burner will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${INPUTS}`.
Command line pattern	

3.5.4.1 S08 Burner > Output > Configure S-Record

Use this panel to control how the burner generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the general options for HCS08 configure S-Record.

Table 3-84. Tool Settings - Burner > Output > Configure S-Record Options

Option	Description
DIsable all (-Ns)	Disables generation of all start (S0) and end records (S7, S8, or S9)
No path in S0-record	Removes the path (if present) from the file name in the S0 record
No S9-record	Disables generation of S9-record
No S8-record	Disables generation of S8-record
No S7-record	Disables generation of S7-record
No S0-record	Disables generation of S0-record

3.5.4.2 S08 Burner > Input

Use this panel to specify the execute command file of the Burner input.

The following table lists and describes the inputs options for burner.

Table 3-85. Tool Settings - Burner > Input Options

Option	Description
Execute command file	This option causes the Burner to execute a Batch Burner command file (usual extension is .bbl).

3.5.4.3 S08 Burner > Host

Use this panel to specify the host settings of the RS08.

The following table lists and describes the memory model options for RS08.

Table 3-86. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.5.4.4 S08 Burner > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-87. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.

Table continues on the next page...

Table 3-87. Tool Settings - Messages Options (continued)

Option	Description
Create err.log Error file	Using this option, the Burner uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Burner with additional arguments (for example, files and Burner options). If you start the Burner with arguments (for example, from the Make Tool or with the ` %f' argument from the CodeWright IDE), the Burner compiles the files in a batch mode. No Burner window is visible and the Burner terminates after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Burner creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.

Table continues on the next page...

Table 3-87. Tool Settings - Messages Options (continued)

Option	Description
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.5.4.4.1 S08 Burner > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-88. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.5.4.5 S08 Burner > General

Use this panel to specify the general linker behavior.

The following table lists and describes the general burner options for **HCS08.**

Table 3-89. Tool Settings - Burner > General Options

Option	Description
Other flags	Specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI. Default value is: `-WmsgSd1100 -WmsgSd1912`

3.5.5 RS08 Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the RS08 Compiler tree control includes the general and the file search path settings.

The following table lists and describes the compiler options for RS08..

Table 3-90. Tool Settings - Compiler Options

Option	Description
Command	Shows the location of the compiler executable file. Default value is : " `${HC08Tools}/crs08.exe`". You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the compiler will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `{COMMAND} ${FLAGS}${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.

3.5.5.1 RS08 Compiler > Output

Use this panel to control how the compiler generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the output options for RS08 compiler.

Table 3-91. Tool Settings - RS08 Compiler > Output Options

Option	Description
Allocate CONST objects in ROM (-Cc)	Check to enables the Compiler assign const objects into the ROM_VAR segment, which the parameter file assigns to a ROM section.
Encrypt File (e.g. %f.e%e) (-Eencrypt)	Encrypts using the given key with the -Ekey: Encryption Key option.
Encryption key (-EKey)	Encrypt files with the given key number (-Eencrypt option).The default encryption key is 0. Using this default is not recommended.
Object File Format	Defines the object file format.
Generate Assembler Include File (e.g. %f.inc) (-La)	Enables the Compiler to generate an assembler include file when the CREATE_ASM_LISTING pragma occurs. The name of the created file is specified by this option. If no name is specified, a default of %f.inc is taken. To put the file into the directory specified by the TEXTPATH: Text File Path environment variable, use the option -la=%n.inc. The %f option already contains the path of the source file. When %f is used, the generated file is in the same directory as the source file. The content of all modifiers refers to the main input file and not to the actual header file. The main input file is the one specified on the command line.
Generate Listing File (e.g. %n.lst) (-Lasm)	Enables the Compiler to generate an assembler listing file directly. The Compiler also prints all assembler-generated instructions to this file. The option specifies the name of the file. If no name is specified, the Compiler takes a default of %n.lst. If the resulting filename contains no path information the Compiler uses the TEXTPATH: Text File Path environment variable. The syntax does not always conform with the inline assembler or the assembler syntax. Therefore, use this option only to review the generated code. It cannot currently be used to generate a file for assembly.
Log predefined defines to file (e.g. predef.h) (-Ldf)	Enables the Compiler to generate a text file that contains a list of the compiler-defined #define. The default filename is predef.h, but may be changed (e.g., -Ldf="myfile.h"). The file is generated in the directory specified by the TEXTPATH: Text File Path environment variable. The defines written to this file depend on the actual Compiler option settings (e.g., type size settings or ANSI compliance). Note: The defines specified by the command line (-D: Macro Definition option) are not included. This option may be very useful for SQA. With this option it is possible to document every #define which was used to compile all sources. Note: This option only has an effect if a file is compiled. This option is unusable if you are not compiling a file.
List of included files to `.inc' file (-Li)	Enables the Compiler to generate a text file which contains a list of the #include files specified in the source. This text file shares the same name as the source file but with the extension, *.inc. The files are stored in the path specified by the TEXTPATH: Text File Path environment variable. The generated file may be used in make files.

Table continues on the next page...

Table 3-91. Tool Settings - RS08 Compiler > Output Options (continued)

Option	Description
Write statistic output to file (e.g. logfile.txt) (-Ll)	Enables the Compiler append statistical information about the compilation session to the specified file. The information includes Compiler options, code size (in bytes), stack usage (in bytes) and compilation time (in seconds) for each procedure of the compiled file. The Compiler appends the information to the specified filename (or the file make.txt, if no argument given). Set the TEXTPATH: Text File Path environment variable to store the file into the path specified by the environment variable. Otherwise the Compiler stores the file in the current directory.
List of included files in make format (e.g. logfile.txt)(-Lm)	This option causes the Compiler to generate a text file which contains a list of the #include files specified in the source. The generated list is in a make format. The -Lm option is useful when creating make files. The output from several source files may be copied and grouped into one make file. The generated list is in the make format. The filename does not include the path. After each entry, an empty line is added. The information is appended to the specified filename (or the make.txt file, if no argument is given).
Append object file name to list (e.g. obklist.txt)(-Lo)	This option causes the Compiler to append the object filename to the list in the specified file.The information is appended to the specified filename (or the file make.txt file, if no argument given).
Preprocessor output (e.g. %n.pre)(-Lp)	This option causes the Compiler to generate a text file which contains the preprocessor's output. If no filename is specified, the text file shares the same name as the source file but with the extension, *.PRE (%n.pre). The TEXTPATH environment variable is used to store the preprocessor file.
Strip path information	Check to enable the compiler remove both unreferenced path reference from your program. This reduces your program's memory footprint.

3.5.5.1.1 RS08 Compiler > Output > Configure Listing File

Use this panel to configure the listing files for the RS08 Compiler to generate output.

The following table lists and describes the Configure Listing FIle options for RS08 compiler.

Table 3-92. Tool Settings - RS08 Compiler > Output > Configure Listing File Options

Option	Description
Disable all (-Lasmc)	This option configures the output format of the listing file generated with the Generate Listing File option. The addresses, the hex bytes, and the instructions are selectively switched off.

Table continues on the next page...

Table 3-92. Tool Settings - RS08 Compiler > Output > Configure Listing File Options (continued)

Option	Description
Do not write cycle information (-Lasmc=y)	This option switches off the cycle information from the output format of the listing file.
Do not write compiler version (-Lasmc=v)	This option switches off the compiler version from the output format of the listing file.
Do not write the source code (-Lasmc=s)	This option switches off the source code from the output format of the listing file.
Do not write source prolog (-Lasmc=p)	This option switches off the source prolog from the output format of the listing file.
Do not write the instruction (-Lasmc=i)	This option switches off the instruction from the output format of the listing file.
Do not write the function header (-Lasmc=h)	This option switches off the function header from the output format of the listing file.
Do not write source epilog (-Lasmc=e)	This option switches off the source epilog from the output format of the listing file.
Do not write the code (-Lasmc=c)	This option switches off the code from the output format of the listing file.
Do not write the address (-Lasmc=a)	This option switches off the address from the output format of the listing file.

3.5.5.1.2 RS08 Compiler > Output > Configuration for list of included files in make format

Use this panel to configure the list of included files in make format for the RS08 Compiler to generate the output.

The following table lists and describes the configuration for list of included files in make format options for RS08 compiler.

Table 3-93. Tool Settings - RS08 Compiler > Output > Configure Listing File Options

Option	Description
Disable all (-LmCfg)	This option is used when configuring the List of Included Files in Make Format (-Lm) option. The -LmCfg option is operative only if the -Lm option is also used. The -Lm option produces the `dependency' information for a make file.
Unix style paths (-LmCfg=x)	Use this option to writes the path names in Unix style.
Update information (-LmCfg=u)	This option updates the information in the output file. If the file does not exist, the Compiler creates the file. If the file exists and the current information is not yet in the file, the Compiler appends the information to the file. If the information is already present, the Compiler updates the information. This allows you to specify this suboption for each compilation ensuring that the make dependency file is always up to date.

Table continues on the next page...

Table 3-93. Tool Settings - RS08 Compiler > Output > Configure Listing File Options (continued)

Option	Description
Write path of object file (-LmCfg=o)	This option writes the full name of the target object file.
Write path of main file (-LmCfg=m)	This option writes the full path of the compiled file. This is necessary when there are files with the same name in different directories.
Use line continuation (-LmCfg=l)	This option uses line continuation for each single entry in the dependency list. This improves readability.
Write path of included file (-LmCfg=i)	This option writes the full path of all included files in the dependency list.

3.5.5.2 RS08 Compiler > Input

Use this panel to specify file search paths and any additional include files the RS08 Compiler should use. You can specify multiple search paths and the order in which you want to perform the search.

The IDE first looks for an include file in the current directory, or the directory that you specify in the `INCLUDE` directive. If the IDE does not find the file, it continues searching the paths shown in this panel. The IDE keeps searching paths until it finds the `#include` file or finishes searching the last path at the bottom of the Include File Search Paths list. The IDE appends to each path the string that you specify in the `INCLUDE` directive.

NOTE

> The IDE displays an error message if a header file is in a different directory from the referencing source file. Sometimes, the IDE also displays an error message if a header file is in the same directory as the referencing source file.

For example, if you see the message `Could not open source file myfile.h`, you must add the path for `myfile.h` to this panel.

The following table lists and describes the input options for RS08 Compiler.

Table 3-94. Tool Settings - RS08 Compiler > Input Options

Option	Description
Filenames are clipped to DOS length (-!)	The filenames are clipped to DOS length (eight characters), when compiling files from MS-DOS file system.
Include File Path (-I)	Specify, delete, or rearrange file search paths.

Table continues on the next page...

Table 3-94. Tool Settings - RS08 Compiler > Input Options (continued)

Option	Description
Recursive Include File Path (-Ir)	Appends a recursive access path to the current #include list. This command is global. **Syntax**`-ir pathpath` The recursive access path to append.
Additional Include Files (-AddInd)	Specify, delete, or rearrange paths to search any additional #include files.
Include files only once	Check to include every header file only once; duplicates are ignored.

The following table lists and describes the toolbar buttons that help work with the file paths.

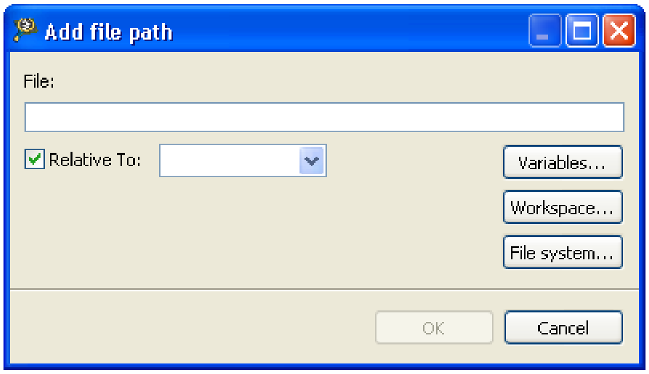
Table 3-95. Include File Path (-I) Toolbar Buttons

Button	Description
➕	Add - Click to open the Add directory path dialog box and specify location of the library you want to add.
✖	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected path.
⇧	Move up - Click to move the selected path one position higher in the list.
⇩	Move down - Click to move the selected path one position lower in the list.

The following figure lists and describes the toolbar buttons that help work with the search paths.

Table 3-96. Additional Include Files (-AddIncl) Toolbar Buttons

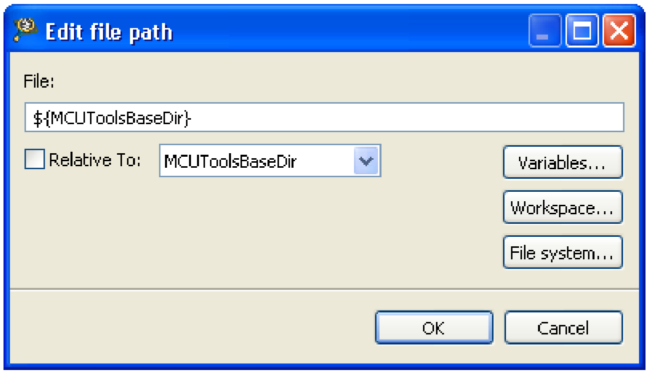
Button	Description
➕	Add - Click to open the Add directory path dialog box and specify location of the library you want to add.
✖	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected path.
⇧	Move up - Click to move the selected path one position higher in the list.
⇩	Move down - Click to move the selected path one position lower in the list.



File:

☑ Relative To:

Figure 3-17. Tool Settings - RS08 Compiler > Input - Add file path Dialog Box



File: ${MCUToolsBaseDir}

☐ Relative To: MCUToolsBaseDir

Figure 3-18. Tool Settings - RS08 Compiler > Input - Edit file path Dialog Box

The buttons in the **Add file path and Edit file path** dialog boxes help work with the paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **File Selection** dialog box and specify the path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Open** dialog box and specify the path. The resulting path appears in the appropriate list.

3.5.5.3 RS08 Compiler > Language

Use this panel to specify code- and symbol-generation options for the RS08 Compiler.

The following table lists and describes the language options for RS08.

Table 3-97. Tool Settings - RS08 Compiler > Language Options

Option	Description
Strict ANSI	Check if you want the C compiler to operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ISO specification to all input files. This setting is equivalent to specifying the `- ansi` command-line option. The compiler issues a warning for each ANSI/ISO extension it finds.
C++	With this option enabled, the Compiler behaves as a C++ Compiler. You can select between three different types of C++: • **Full C++ (-C++f) -** Supports the whole C++ language. • **Embedded C++ (-C++e)** - Supports a constant subset of the C++ language. EC++ does not support inefficient things like templates, multiple inheritance, virtual base classes and exception handling. • **CompactC++ (-C++c) -** Supports a configurable subset of the C++ language. You can configure this subset with the option -Cn. • **No C++** - If the option is not set, the Compiler behaves as an ANSI-C Compiler. If the option is enabled and the source file name extension is *.c, the Compiler behaves as a C++ Compiler. If the option is not set, but the source filename extension is .cpp or .cxx, the Compiler behaves as if the -C++f option were set.
Cosmic compatibility mode for space modifiers @near, @far, and @tiny (-Ccx)	Check to allow Cosmic style @near, @far and @tiny space modifiers as well as @interrupt in your C code. The -ANSI option must be switched off. It is not necessary to remove the Cosmic space modifiers from your application code. There is no need to place the objects to sections addressable by the Cosmic space modifiers. The following is done when a Cosmic modifier is parsed: The objects declared with the space modifier are always allocated in a special Cosmic compatibility (_CX) section (regardless of which section pragma is set) depending on the space modifier, on the const qualifier or if it is a function or a variable. Space modifiers on the left hand side of a pointer declaration specify the pointer type and pointer size, depending on the target.
Bigraph and trigraph support (-Ci)	Check to replace certain unavailable tokens with the equivalent keywords.
C++ comments in ANSI-C (-Cppc)	Check to allow C++ comments.
Propagate const and volatile qualifiers for structs (-Cq)	Check to propagate const and volatile qualifiers for structures. If all members of a structure are constant or volatile, the structure itself is constant or volatile. If the structure is declared as constant or volatile, all its members are constant or volatile, respectively.
Conversion from `const T*' to `T*' (-Ec)	Check to enable this non-ANSI compliant extension allows the compiler to treat a pointer to a constant type like a pointer to the non-constant equivalent of the type. Earlier Compilers did not check a store to a constant object through a pointer. This option is useful when compiling older source code.

Table continues on the next page...

Table 3-97. Tool Settings - RS08 Compiler > Language Options (continued)

Option	Description
Do not pre-process escape sequences in strings with absoluted DOS paths (-Pe)	If escape sequences are used in macros, they are handled in an include directive similar to the way they are handled in a printf() instruction. If the -Pe option is used, escape sequences are ignored in strings that contain a DOS drive letter ('a - 'z', 'A' - 'Z') followed by a colon ':' and a backslash '\'. When the -Pe option is enabled, the Compiler handles strings in include directives differently from other strings. Escape sequences in include directive strings are not evaluated.

3.5.5.3.1 RS08 Compiler > Language > CompactC++ features

Use this panel to select compact C++ features of RS08 compiler.

The following table lists and describes the compactC++ options for HCS08.

Table 3-98. Tool Settings - RS08 Compiler > Language > CompactC++ Features Options

Option	Description
Disable all compactC++ features (-Cn)	If the -C++ option is enabled, you can disable the compactC++ features. • Vf : Virtual functions are not allowed. Avoid having virtual tables that consume a lot of memory. • Tpl : Templates are not allowed. Avoid having many generated functions perform similar operations. • Ptm : Pointer to member not allowed. Avoid having pointer-to-member objects that consume a lot of memory. • Mih : Multiple inheritance is not allowed. Avoid having complex class hierarchies. Because virtual base classes are logical only when used with multiple inheritance, they are also not allowed. • Ctr : The C++ Compiler can generate several kinds of functions, if necessary: - Default Constructor - Copy Constructor - Destructor - Assignment operator

Table continues on the next page...

Table 3-98. Tool Settings - RS08 Compiler > Language > CompactC++ Features Options (continued)

Option	Description
	With this option enabled, the Compiler does not create those functions. This is useful when compiling C sources with the C++ Compiler, assuming you do not want C structures to acquire member functions. • Cpr : Class parameters and class returns are not allowed. Avoid overhead with Copy Constructor and Destructor calls when passing parameters, and passing return values of class type.
Do not allow virtual functions (-Cn=Vf)	Virtual functions are not allowed. Avoid having virtual tables that consume a lot of memory
Do not allow templates (-Cn=Tpl)	Templates are not allowed. Avoid having many generated functions perform similar operations.
Do not allow pointer to member (-Cn=Ptm)	Pointer to member not allowed. Avoid having pointer-to-member objects that consume a lot of memory.
Do not allow multiple inheritance and virtual base classes (-Cn=Mih)	Multiple inheritance is not allowed. Avoid having complex class hierarchies. Because virtual base classes are logical only when used with multiple inheritance, they are also not allowed.
Do not create compiler defined functions (-Cn=Ctr)	The C++ Compiler can generate several kinds of functions, if necessary: • Default Constructor • Copy Constructor • Destructor • Assignment operator With this option enabled, the Compiler does not create those functions. This is useful when compiling C sources with the C ++ Compiler, assuming you do not want C structures to acquire member functions.
Do not allow class parameters and class returns (-Cn=Ctr)	Class parameters and class returns are not allowed. Avoid overhead with Copy Constructor and Destructor calls when passing parameters, and passing return values of class type.

3.5.5.4 RS08 Compiler > Host

Use this panel to specify the host settings of the RS08.

The following table lists and describes the memory model options for RS08.

Table 3-99. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • -ViewWindow, the application appears with its normal window. • -ViewMin the application appears as an icon in the task bar. • -ViewMax, the application appears maximized (filling the whole screen). • -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.5.5.5 RS08 Compiler > Code Generation

Use this panel to specify code- and symbol-generation options for the RS08 Compiler

The following table lists and describes the code generation options for RS08 compiler.

Table 3-100. Tool Settings - RS08 Compiler > Code Generation Options

Option	Description
Bit field byte allocation (-BfaB[MS|LS])	Normally, bits in byte bitfields are allocated from the least significant bit to the most significant bit. This produces less code overhead if a byte bitfield is allocated only partially.
Bit field gap limit (-BfaGapLimitBits)	Check to affect the maximum allowable number of gap bits. The bitfield allocation tries to avoid crossing a byte boundary whenever possible. To optimize accesses, the compiler may insert some padding or gap bits.

Table continues on the next page...

Table 3-100. Tool Settings - RS08 Compiler > Code Generation Options (continued)

Option	Description
Bit field type size reduction	This option is configurable whether or not the compiler uses type-size reduction for bitfields. Type-size reduction means that the compiler can reduce the type of an int bitfield to a char bitfield if it fits into a character. This allows the compiler to allocate memory only for one byte instead of for an integer. Options are: • **Enabled (-BfsTSRON)** • **Disabled (-BfsTSOFF)**
Maximum load factor for switch tables (0-100) (-CswMaxLF)	Allows changing the default strategy of the Compiler to use tables for switch statements; is only available if the compiler supports switch tables.
Minimum number of labels for switch tables (-CswMinLB)	Allows changing the default strategy of the Compiler using tables for switch statements; is only available if the compiler supports switch tables.
Minimum load factor for switch tables (0-100) (-CswMinLF)	Allows the Compiler to use tables for switch statements; is only available if the compiler supports switch tables.
Minimum number of labels for switch search tables (-CswMinSLB)	Allows the Compiler to use tables for switch statements. Using a search table improves code density, but the execution time increases. Every time an entry in a search table must be found, all previous entries must be checked first. For a dense table, the right offset is computed and accessed. In addition, note that all backends implement search tables (if at all) by using a complex runtime routine. This may make debugging more complex.
Switch off code generation (-Cx)	Disables the code generation process of the Compiler. No object code is generated, though the Compiler performs a syntactical check of the source code. This allows a quick test if the Compiler accepts the source without errors.
Do not use CLR for volatile variables in the direct page (-NoClrVol)	Inhibits the use of CLR for volatile variables in the direct page. The CLR instruction on HC08 has a read cycle. This may lead to unwanted lateral effects (e.g. if the variable is mapped over a hardware register).
Qualifier for virtual table pointers (-Qvtp)	Using a virtual function in C++ requires an additional pointer to virtual function tables. The Compiler cannot access the pointer and generates the pointer in every class object when virtual function tables are associated.
Use IEEE32 for double	Check to use IEEE32 for doubles instead of IEEE64 (default).
Specify the address of the Interrupt Exit address register (-IEA)	Specifies the address of the interrupt exit address register. By default, it is 0x200.
Specify the address of the System Interrupt Pending 2 register (-SIP2)	Specifies the address of the System Interrupt Pending 2 register. By default, it is set to 0x1D.

3.5.5.6 RS08 Compiler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-101. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Compiler uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Compiler with additional arguments (for example, files and Compiler options). If you start the Compiler with arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the Compiler compiles the files in a batch mode. No Compiler window is visible and the Compiler terminates after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.

Table continues on the next page...

Table 3-101. Tool Settings - Messages Options (continued)

Option	Description
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Compiler creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.5.5.6.1 RS08 Compiler > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging. The following table lists and describes the message options.

Table 3-102. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.5.5.7 RS08 Compiler > Preprocessor

Use this panel to specify preprocessor behavior and define macros.

The following table lists and describes the preprocessor options for RS08 Compiler.

Table 3-103. Tool Settings - RS08 Compiler > Preprocessor Options

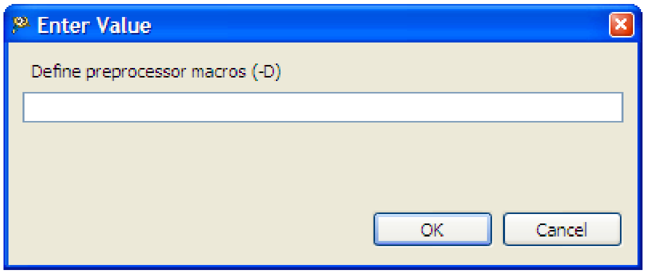
Option	Description
Define preprocessor macros (-D)	Define, delete, or rearrange preprocessor macros. You can specify multiple macros and change the order in which the IDE uses the macros. Define preprocessor macros and optionally assign their values. This setting is equivalent to specifying the `-D name[=value]` command-line option. To assign a value, use the equal sign (=) with no white space. For example, this syntax defines a preprocessor value named `EXTENDED_FEATURE` and assigns `ON` as its value: `EXTENDED_FEATURE=ON` Note that if you do not assign a value to the macro, the shell assigns a default value of 1.

The following table lists and describes the toolbar buttons that help work with preprocessor macro definitions.

Table 3-104. Define Preprocessor Macros Toolbar Buttons

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the path/macro.
✖	Delete - Click to delete the selected path/macro. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected path/macro.
⇧	Move up - Click to move the selected path/macro one position higher in the list.
⇩	Move down - Click to move the selected path/macro one position lower in the list

The following figure shows the Enter Value dialog box.

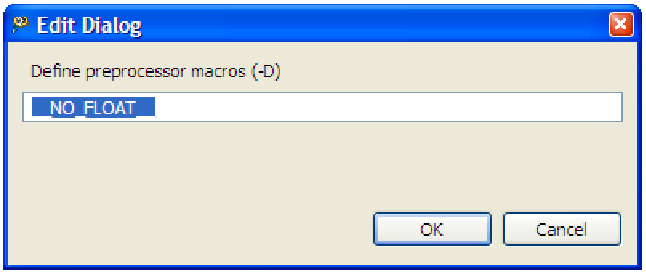


Enter Value

Define preprocessor macros (-D)

Figure 3-19. Tool Settings - RS08 Compiler > Preprocessor - Enter Value Dialog Box

The following figure shows the Edit Dialog dialog box.



Edit Dialog

Define preprocessor macros (-D)

NO_FLOAT

Figure 3-20. Tool Settings - RS08 Compiler > Preprocessor - Edit Dialog Box

The buttons in the **Enter Value and Edit** dialog boxes help work with the preprocessor macros.

- OK **-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.5.5.8 RS08 Compiler > Type Sizes

Use this panel to specify the available data type size options for the RS08 Compiler.

The following table lists and describes the type size options for RS08 Compiler.

Table 3-105. Tool Settings - RS08 Compiler > Type Sizes

Option	Description
char	Selects the size of the char type. Options are: • Default (unsigned 8bit) • unsigned 8bit (-TuCC1) • signed 8bit (-TsCC1) • signed 16bit (-TsCC2) • signed 32bit (-TsCC4)
short	Selects the size of the short type. Options are: • Default (16bit) • signed 8bit (-TS1)

Table continues on the next page...

Table 3-105. Tool Settings - RS08 Compiler > Type Sizes (continued)

Option	Description
	• signed 16bit (-TS2) • signed 32bit (-TS4)
int	Selects the size of the int type. Options are: • Default (16bit) • signed 8bit (-TI1) • signed 16bit (-TI2) • signed 32bit (-TI4)
long	Selects the size of the long type. Options are: • Default (32bit) • signed 8bit (-TL1) • signed 16bit (-TL2) • signed 32bit (-TL4)
long long	Selects the size of the long long type. Options are: • Default (32bit) • signed 8bit (-TLL1) • signed 16bit (-TLL2) • signed 32bit (-TLL4)
enum	Selects the size of the enum type. Options are: • Default (signed 16bit) • signed 8bit (-TE1sE) • signed 16bit (-TE2sE) • signed 32bit (-TE4sE) • unsigned 8bit (-TE1uE)
float	Selects the size of the float type. Options are: • Default (IEEE32) • IEEE32
double	Selects the size of the double type. Options are: • Default (IEEE32) • IEEE32
long double	Selects the size of the long double type. Options are: • Default (IEEE32) • IEEE32
long long double	Selects the size of the long long double type. Options are: • Default (IEEE32) • IEEE32

3.5.5.9 RS08 Compiler > General

Use this panel to specify other flags for the RS08 Compiler to use.

The following table lists and describes the general options for RS08 compiler.

Table 3-106. Tool Settings - RS08 Compiler > General Options

Option	Description
Other flags	Specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.

3.5.5.10 RS08 Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the optimization options for RS08 compiler.

Table 3-107. Tool Settings - RS08 Compiler > Optimization Options

Option	Description
No integral promotion on characters (-Cni)	Enhances character operation code density by omitting integral promotion. This option enables behavior that is not ANSI-C compliant. Code generated with this option set does not conform to ANSI standards. Code compiled with this option is not portable. Using this option is not recommended in most cases.
Loop unrolling (i[number]) (-Cu)	Enables loop unrolling with the following restrictions: • Only simple for statements are unrolled, e.g., for (i=0; i<10; i++) • Initialization and test of the loop counter must be done with a constant. • Only <, >, <=, >= are permitted in a condition. • Only ++ or -- are allowed for the loop variable increment or decrement. • The loop counter must be integral. • No change of the loop counter is allowed within the loop. • The loop counter must not be used on the left side of an assignment. • No address operator (&) is allowed on the loop counter within the loop. • Only small loops are unrolled: Loops with few statements within the loop. Loops with fewer than 16 increments or decrements of the loop counter. The bound may be changed with the optional argument = i<number>.

Table continues on the next page...

Table 3-107. Tool Settings - RS08 Compiler > Optimization Options (continued)

Option	Description
	The -Cu=i20 option unrolls loops with a maximum of 20 iterations.
Main Optimize Target: Optimize for	There are various points where the Compiler has to select between two possibilities: it can either generate fast, but large code, or small but slower code. The Compiler generally optimizes on code size. It often has to decide between a runtime routine or an expanded code. The programmer can decide whether to select between the slower and shorter or the faster and longer code sequence by setting a command line switch. • The **Code Size (-Os)** option directs the Compiler to optimize the code for smaller code size. The Compiler trades faster-larger code for slower-smaller code. • The **Execution Time (-Ot)** option directs the Compiler to optimize the code for faster execution time. The Compiler replaces slower/smaller code with faster/ larger code. This option only affects some special code sequences. This option has to be set together with other optimization options (e.g., register optimization) to get best results.
Create sub-functions with common code	Performs the reverse of inlining. It detects common code parts in the generated code. The Compiler moves the common code to a different place and replaces all occurrences with a JSR to the moved code. At the end of the common code, the Compiler inserts an RTS instruction. The Compiler increases all SP uses by an address size. This optimization takes care of stack allocation, control flow, and of functions having arguments on the stack. Inline assembler code is never treated as common code. Options are: • **Default** • **Disable (-Onf)** • **Enable (-Of)**
Alias analysis options	These four different options allow the programmer to control the alias behavior of the compiler. The option -oaaddr is the default because it is safe for all C programs. Use option -oaansi if the source code follows the ANSI C99 alias rules. If objects with different types never overlap in your program, use option -oatype. If your program doesn't have aliases at all, use option -oanone (or the ICG option -ona, which is supported for compatibility reasons).
Generate always near calls (-Obsr)	This option forces the compiler to always generate near calls, i.e. use BSR instruction instead of a JSR in order to reduce code size. Without this option the compiler checks the range of the call to determine if a BSR can be generated instead of a JSR.
Dynamic options configuration for functions (-OdocF)	Allows the Compiler to select from a set of options to reach the smallest code size for every function. Without this feature, you must set fixed Compiler switches over the whole compilation unit. With this feature, the Compiler finds the best option combination from a user-defined set for every function.

Table continues on the next page...

Table 3-107. Tool Settings - RS08 Compiler > Optimization Options (continued)

Option	Description
Inlining (C[n] or OFF) (-Oi)	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. Using the -Oi=c0 option switches off inlining. Functions marked with the #pragma INLINE are still inlined. To disable inlining, use the -Oi=OFF option.
Disable alias checking (-Ona)	Prevents the Compiler from redefining these variables, which lets you reuse already-loaded variables or equivalent constants. Use this option only when you are sure no real writes of aliases to a variable memory location will occur.
Disable branch optimizer (-OnB)	Disables all branch optimizations.
Do generate copy down information for zero values (-OnCopyDown)	Restricts the compiler from generating a copy down for i. The initialization with zero optimization shown for the arr array only works in the HIWARE format. The ELF format requires initializing the whole array to zero.
Disable CONST variable by constant replacement (-OnCsfVar)	Lets you switch OFF the replacement of CONST variable by the constant value.
Disable peephole optimization (-OnP)	Disables the whole peephole optimizer. To disable only a single peephole optimization, use the optional syntax -OnP=<char>.
Disable code generation for NULL Pointer to Member check (-OnPMNC)	Before assigning a pointer to a member in C++, you must ensure that the pointer to the member is not NULL in order to generate correct and safe code. In embedded systems development, the difficulty becomes generating the denser code while avoiding overhead whenever possible (this NULL check code is a good example). This option enables you to switch off the code generation for the NULL check.
Large return value type	Compiler supports this option even though returning a 'large' return value may be not as efficient as using an additional pointer. The Compiler introduces an additional parameter for the return value if the return value cannot be passed in registers. Options are: • **Default** • **Large return value pointer, always with temporary (-Rpt)** • **Large return value pointer and temporary elimination (-Rpe)**
Disable far to near optimization	Disables the JSR to BSR optimization. The compiler checks the range of the call to determine if a BSR can be generated instead of a JSR. If -Onbsr is used this optimization will be disabled.
Disable reload from register optimization	Disables the low level register trace optimizations. If you use the option the code becomes more readable, but less optimal.
Disable tail call optimizations	Allows the compiler to remove all the entry and exit code from the current function.By default, the compiler replaces trailing calls (JSR/BSR) with JMP instructions if the function does not contain any other function calls.
Reuse locals of stack frame	Instructs the compiler to reuse the location of local variables/ temporaries whenever possible. When used, the compiler analyzes which local variables are alive simultaneously.

Table 3-107. Tool Settings - RS08 Compiler > Optimization Options

Option	Description
	Based on that analysis the compiler selects the best memory layout for for variables. Two or more variables may end up sharing the same memory location.

3.5.5.10.1 RS08 Compiler > Optimization > Mid level optimizations

Use this panel to configure the Mid level optimization options for the RS08 compiler.

The following table lists and describes the Mid level optimizations options for RS08 compiler.

Table 3-108. Tool Settings - RS08 Compiler > Optimization > Mid level optimizations

Option	Description
Disable all optimizations (-Od)	The backend of this compiler is based on the second generation intermediate code generator (SICG). All intermediate language and processor independent optimizations (cf. NULLSTONE) are performed by the SICG optimizer using the powerful static single assignment form (SSA form). The optimizations are switched off using -od. Currently four optimizations are implemented. This option disables all the optimizations.
Disable mid level loop induction variable elimination (-Od=g)	This option disables all the mid level loop induction variable elimination.
Disable mid level code motion (-Od=f)	This option disables all the mid level code motion.
Disable mid level instruction combination (-Od=e)	This option disables all the mid level instruction combination.
Disable mid level removing dead assignments (-Od=d)	This option disables removing dead assignments only.
Disable mid level common subexpression elimination (-Od=c)	This option disables removing dead assignments and CSE.
Disable mid level constant propagation (-Od=b)	This option disables mid level constant propagation.
Disable mid level copy propagation (-Od=a)	This option disables mid level copy propagation

3.5.5.10.2 RS08 Compiler > Optimization > Mid level branch optimizations

Use this option to specify the mid level branch optimization options.

The following table lists and describes the Mid level branch optimizations options for RS08 compiler.

Table 3-109. Tool Settings - RS08 Compiler > Optimization > Mid level branch optimizations

Option	Description
Disable all optimizations (-Odb)	This option disables branch optimizations on the SSA form based on control flows. Label rearranging sorts all labels of the control flow to generate a minimum amount of branches.
Disable mid level loop hoisting (-Odb=c)	This option disables mid level loop hoisting.
Disable mid level branch tail merging (-Odb=b)	This option disables only branch tail merging.
Disable mid level label rearranging (-Odb=a)	This option disables mid level label rearranging.

3.5.5.10.3 RS08 Compiler > Optimization > Tree optimizer

The Compiler contains a special optimizer which optimizes the internal tree data structure. This tree data structure holds the semantic of the program and represents the parsed statements and expressions.

This option disables the tree optimizer. This may be useful for debugging and for forcing the Compiler to produce `straightforward' code.

Use this panel to configure the tree optimizer options for the RS08 compiler.

The following table lists and describes the Tree optimizer options for RS08 compiler.

Table 3-110. Tool Settings - RS08 Compiler > Optimization > Tree optimizer

Option	Description
Disable all optimizations (-Ont)	Disable all the optimizations.
Disable bit neg optimization (-Ont=~)	Disable optimization of `~~i' into `i'.
Disable bit or optimization (-Ont=I)	Disable optimization of `il0xffff' into `0xffff'.
Disable exor optimization (-Ont=^)	Disable optimization of `i^0' into `i'.
Disable if optimization (-Ont=w)	Disable optimization of `if (1) i = 0;' into `i = 0;'.
Disable do optimization (-Ont=v)	Disable optimization of `do ... while(0) into `...'.
Disable while optimization (-Ont=u)	Disable optimization of `while(1) ...;' into `...;'.
Disable for optimization (-Ont=t)	Disable optimization of `for(;;) ...' into `while(1) ...'.
Disable indirect optimization (-Ont=s)	Disable optimization of `*&i' into `i'.
Disable 16-32 relative optimization (-Ont=r)	Disable optimization of `L<=4' into 16-bit compares if 16-bit compares are better.
Disable 16-32 compare optimization (-Ont=q)	Reduction of long compares into int compares if int compares are better: (-Ont=q to disable it).
Disable cut optimization (-Ont=p)	Disable optimization of `(char)(long)i' into `(char)i'.

Table continues on the next page...

Table 3-110. Tool Settings - RS08 Compiler > Optimization > Tree optimizer (continued)

Option	Description
Disable cast optimization (-Ont=o)	Disable optimization of `(short)(int)L' into `(short)L' if short and int have the same size.
Disable right shift optimization (-Ont=n)	Optimization of shift optimizations (<<, -Ont=n to disable it)
Disable left shift optimization (-Ont=m)	Optimization of shift optimizations (>>, -Ont=m to disable it)
Disable label optimization (-Ont=l)	Disable optimization removal of labels if not used.
Disable transformations for inlining optimization (-Ont=j)	This optimization transforms the syntax tree into an equivalent form in which more inlining cases can be done. This option only has an effect when inlining is enabled.
Disable address optimization (-Ont=i)	Disable optimization of `&*p' into `p'.
Disable unary minus optimization (-Ont=h)	Disable optimization of `-(-i)' into `i'.
Disable compare size optimization (-Ont=g)	Disable optimization of compare size.
Disable condition optimization (-Ont=f)	Disable optimization of `(a==0)' into `(!a)'.
Disable const swap optimization (-Ont=e)	Disable optimization of `2*i' into `i*2'.
Disable binary operation optimization (-Ont=d)	Disable optimization of `us & ui' into `us & (unsigned short) ui'.
Disable compare optimization (-Ont=c)	Disable optimization of `if ((long)i)' into `if (i)'.
Disable constant folding optimization (-Ont=b)	Disable optimization of `3+7' into `10'.
Disable statement optimization (-Ont=a)	Disable optimization of last statement in function if result is not used.
Disable test optimization (-Ont=?)	Disable optimization of `i = (int)(cond ? L1:L2);' into `i = cond ? (int)L1:(int)L2;'.
Disable assign optimization (-Ont=9)	Disable optimization of `i=i;'.
Disable switch optimization (-Ont=8)	Disable optimization of empty switch statement.
Disable extend optimization (-Ont=7)	Disable optimization of `(long)(char)L' into `L'.
Disable or optimization (-Ont=1)	Disable optimization of `a || 0' into `a'.
Disable and optimization (-Ont=0)	Disable optimization of `a && 1' into `a'.
Disable div optimization (-Ont=/)	Disable optimization of `a/1' into `a'.
Disable minus optimization (-Ont=-)	Disable optimization of `a-0' into `a'.
Disable plus optimization (-Ont=+)	Disable optimization of `a+0' into `a'.
Disable mul optimization (-Ont=*)	Disable optimization of `a*1' into `a'.
Disable bit and optimization (-Ont=)	Disable optimization of `a&0' into `0'.
Disable mod optimization (-Ont=%)	Disable optimization of `a%1' into `0'.

3.5.5.10.4 RS08 Compiler > Optimization > Optimize Library Function

This option enables the compiler to optimize specific known library functions to reduce execution time. The Compiler frequently uses small functions such as strcpy(), strcmp(), and so forth. Use this panel to configure the optimize library function options for the RS08 compiler.

The following table lists and describes the Mid level branch optimizations options for RS08 compiler.

Table 3-111. Tool Settings - RS08 Compiler > Optimization > Mid level branch optimizations

Option	Description
Apply all optimizations (-OiLib)	This option applies all the optimizations.
shifts left of 1 (-OiLib=g)	This option replace shifts left of 1 by array lookup.
memcpy (-OiLib=f)	This option inline calls to the memcpy() function.
memset (-OiLib=e)	This option inline calls to the memset() function.
fabs/fabsf (-OiLib=d)	This option inline calls to the fabs() or fabsf() functions.
strlen (-OiLib=b)	This option inline calls to the strlen() function.

3.5.6 RS08 Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler.

The following table lists and describes the assembler options for RS08.

Table 3-112. Tool Settings - Assembler Options

Option	Description
Command	Shows the location of the assembler executable file. Default value is: `"${HC08Tools}/ahc08.exe"`. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${COMMAND} ${FLAGS}-Objn${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.
Command line pattern	

3.5.6.1 RS08 Assembler > Output

Use this panel to control how the assembler generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the output options for RS08 Assembler.

Table 3-113. Tool Settings - RS08 Assembler > Output Options

Option	Description
Object File Format (-F)	Defines the object file format.
Show label statistics (-Li)	Using the -Ll option, the Compiler appends statistical information about the compilation session to the specified file. The information includes Compiler options, code size (in bytes), stack usage (in bytes) and compilation time (in seconds) for each procedure of the compiled file. The Compiler appends the information to the specified filename (or the file make.txt, if no argument given). Set the TEXTPATH: Text File Path environment variable to store the file into the path specified by the environment variable. Otherwise the Compiler stores the file in the current directory.
Generate listing file (e.g. %(TEXTPATH)/%n.lst) (-L)	The -Lasm option causes the Compiler to generate an assembler listing file directly. The Compiler also prints all assembler-generated instructions to this file. The option specifies the name of the file. If no name is specified, the Compiler takes a default of %n.lst. If the resulting filename contains no path information the Compiler uses the TEXTPATH: Text File Path environment variable. The syntax does not always conform with the inline assembler or the assembler syntax. Therefore, use this option only to review the generated code. It cannot currently be used to generate a file for assembly.
Address size in the listing file (-Lasms)	Specifies the size of the addresses displayed in the listing. Options are: • 1 to display addresses as xx • 2 to display addresses as xxxx • 3 to display addresses as xxxxxx • 4 to display addresses asf xxxxxxxx
Do not print macro call in listing file (-Lc)	Specifies whether macro calls encountered in the source code are expanded and appear in the listing file.
Do not print macro definition in listing file (-Ld)	Instructs the Assembler to generate a listing file but not including any macro definitions. The listing file contains macro invocation and expansion lines as well as expanded include files.
Do not print macro expansion in listing file (-Le)	Switches on the generation of the listing file, but macro expansions are not present in the listing file. The listing file contains macro definition and invocation lines as well as expanded include files.
Do not print included files in listing file (-Li)	Switches on the generation of the listing file, but include files are not expanded in the listing file. The listing file contains macro definition, invocation, and expansion lines.

3.5.6.1.1 RS08 Assembler > Output > Configure Listing File

Use this panel to configure the listing file options of RS08 assembler.

The following table lists and describes the Configure Listing File options for RS08 Assembler.

Table 3-114. Tool Settings - RS08 Assembler > Configure Listing File Options

Option	Description
Disable all (-Lasmc)	Print all the columns in the listing file
Do not write the source line (-Lasmc=s)	Do not print source column in the listing file
Do not write the relative line (-Lasmc=r)	Do not print relative column (Rel.) in the listing file
Do not write the macro mark (-Lasmc=m)	Do not print macro mark column in the listing file
Do not write the address (-Lasmc=l)	Do not print address column (Loc) in the listing file
Do not write the location kind (-Lasmc=k)	Do not print the location type column in the listing file
Do not write the include mark column (-Lasmc=i)	Do not print the include mark column in the listing file
Do not write the object code (-Lasmc=c)	Do not print the object code in the listing file
Do not write the absolute line (-Lasmc=a)	Do not print the absolute column (Abs.) in the listing file

3.5.6.2 RS08 Assembler > Input

Use this panel to specify file search paths and any additional include files the RS08 Assembler should use. You can specify multiple search paths and the order in which you want to perform the search.

The following table lists and describes the input options of RS08 assembler.

Table 3-115. Tool Settings - Assembler > Input options

Button	Description
Include file search paths (-I)	Lists the included file search paths.
Case sensitivity on label names (-Ci)	Check to make the label names case sensitive.
Define label (use spaces to separate labels) (-D)	Define labels that have to be included in the RS08 assembler input.
Support for structured types (-Struct)	Check to include the support for structured types.

The following table lists and describes the toolbar buttons that help work with the file search paths.

Table 3-116. Search Paths Toolbar Buttons

Button	Description
+	Add - Click to open the **Add directory path** dialog box and specify the file search path.

Table continues on the next page...

Table 3-116. Search Paths Toolbar Buttons (continued)

Button	Description
✖	Delete - Click to delete the selected file search path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected file search path one position higher in the list.
⇩	Move down - Click to move the selected file search path one position lower in the list.

3.5.6.3 RS08 Assembler > Language

Use this panel to specify code- and symbol-generation options for the RS08 Assembler.

The following table lists and describes the language options for RS08 Assembler.

Table 3-117. Tool Settings - RS08 Assembler > Language Options

Option	Description
Angle brackets for macro arguments grouping (-CMacAngBrack)	Controls whether the < > syntax for macro invocation argument grouping is available. When it is disabled, the Assembler does not recognize the special meaning for < in the macro invocation context. There are cases where the angle brackets are ambiguous. In new code, use the [? ?] syntax instead. Options are: • **Allow** • **Disallow**
Square braces for macro arguments grouping (-CMacBrackets)	Controls the availability of the [? ?] syntax for macro invocation argument grouping. When it is disabled, the Assembler does not recognize the special meaning for [?] in the macro invocation context. Options are: • **Allow** • **Disallow**
Maximum MacroNest nesting (-MacroNest)	Controls how deep macros calls can be nested. Its main purpose is to avoid endless recursive macro invocations.

3.5.6.3.1 RS08 Assembler > Language > Compatibility modes

Use this panel to specify the compatibility modes options of the RS08 assembler.

The following table lists and describes the compatibility mode options for RS08 Assembler.

Table 3-118. Tool Settings - RS08 Assembler > Compatibility modes Options

Option	Description
Select all (-Compat)	Check to enable all compatibility mode options.
Symbol prefixes (-Compat=s)	With this suboption, the Assembler accepts "pgz:" and "byte:" prefixed for symbols in XDEFs and XREFs. They correspond to XREF.B or XDEF.B with the same symbols without the prefix.
Ignore FF character at line start Symbol prefixes (-Compat=f)	With this suboption, an otherwise improper character recognized from feed character is ignored.
Alternate comment rules (-Compat=c)	With this suboption, comments implicitly start when a space is present after the argument list. A special character is not necessary. Be careful with spaces when this option is given because part of the intended arguments may be taken as a comment. However, to avoid accidental comments, the Assembler does issue a warning if such a comment does not start with a "*" or a ";".
Support FOR directive (-Compat=b)	With this suboption, the Assembler supports a FOR - Repeat assembly block assembly directive to generate repeated patterns more easily without having to use recursive macros.
Add some additional directives (-Compat=a)	With this suboption, some additional directives are added for enhanced compatibility. The Assembler actually supports a SECT directive as an alias of the usual SECTION - Declare Relocatable Section assembly directive. The SECT directive takes the section name as its first argument.
Operator != means equal (-Compat==)	The Assembler takes the default value of the != operator as not equal, as it is in the C language. For compatibility, this behavior can be changed to equal with this option. Because of the risks involved with this option for existing code, a message is issued for every != which is treated as equal.
Support $ character in symbols (-Compat=)	With this suboption, the Assembler supports to start identifiers with a $ sign.
Support additional ! symbols (-Compat=!)	The following additional operators are defined when this option is used: • !^: exponentiation • !m: modulo • !@: signed greater or equal • !g: signed greater • !%: signed less or equal • !t: signed less than • !$: unsigned greater or equal • !S: unsigned greater • !&: unsigned less or equal • !l: unsigned less • !n: one complement • !w: low operator • !h: high operator **Note:** The default values for the following ! operators are defined: • !.: binary AND

Table 3-118. Tool Settings - RS08 Assembler > Compatibility modes Options

Option	Description
	- !x: exclusive OR - !+: binary OR

3.5.6.4 RS08 Assembler > Host

Use this panel to specify the host settings of the RS08.

The following table lists and describes the memory model options for RS08.

Table 3-119. Tool Settings - Host

Option	Description
Set environment variable (-Env)	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.
Borrow license feature (-LicBorrow)	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait until a license is available from floating license server (-LicWait)	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	Select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. - -ViewWindow, the application appears with its normal window. - -ViewMin the application appears as an icon in the task bar. - -ViewMax, the application appears maximized (filling the whole screen). - -ViewHidden, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

3.5.6.5 RS08 Assembler > Code Generation

Use this panel to specify the code generation options of the RS08 assembler.

The following table lists and describes the Code Generation options for RS08 assembler.

Table 3-120. Tool Settings - RS08 Assembler > Code Generation Options

Option	Description
Associate debug information to assembly source file (-Asmdbg)	Passes the assembly source file name information to DWARF sections. When the output .abs file is debugged, the actual assembly source file is displayed instead of intermediary <filename>.dbg file.

3.5.6.6 RS08 Assembler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-121. Tool Settings - Messages Options

Option	Description
Don't print INFORMATION messages (-W1)	Inhibits information message reporting. Only warning and error messages are generated.
Don't print INFORMATION or WARNING messages (-W2)	Suppresses all messages of type INFORMATION and WARNING. Only ERROR messages are generated.
Create err.log Error file	Using this option, the Assembler uses a return code to report errors back to the tools. When errors occur, 16-bit window environments use err.log files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the err.log file.
Cut file names to Microsoft format to 8.3 (-Wmsg8x3)	Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT. This option truncates the filename to the 8.3 format.
Set message file format for batch mode	Use this option to start the Assembler with additional arguments (for example, files and Assembler options). If you start the Assembler with arguments (for example, from the Make Tool or with the `%f' argument from the CodeWright IDE), the Assembler compiles the files in a batch mode. No Assembler window is visible and the Assembler terminates after job completion.
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`)

Table continues on the next page...

Table 3-121. Tool Settings - Messages Options (continued)

Option	Description
	• `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for no file information (e.g. %K %d: %m)(-WmsgFonf)	If there is no file information available for a message, then <string> defines the message format string to use.
Message Format for no positioning information (%"%f%e%": %K %d: %m)(-WmsgFonp)	If there is no position information available for a message, then <string> defines the message format string to use.
Create Error Listing File	This option controls whether the Assembler creates an error listing file. The error listing file contains a list of all messages and errors that occur during processing.
Maximum number of error messages (-WmsgNe)	Specify the number of errors allowed until the application stops processing.
Maximum number of information messages (-WmsgNi)	Specify the maximum number of information messages allowed.
Maximum number of warning messages (-WmsgNw)	Specify the maximum number of warnings allowed.
Set messages to Disable	Check to disable user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Set messages to Error	Check to enable messages of the ERROR category.
Set messages to Information	Check to enable messages of the INFORMATION category.
Set messages to Warning	Check to enable messages of the WARNING category.

3.5.6.6.1 RS08 Assembler > Messages > Disable user messages

Use this panel to specify whether to generate symbolic information for debugging the The following table lists and describes the message options.

Table 3-122. Tool Settings - Disable user messages Options

Option	Description
Disable all messages	Check to disable all the user messages and allow only the normal message categories (WARNING, INFORMATION, ERROR, or FATAL); reduces the number of messages, and simplifies the error parsing of other tools.
Display type of messages (-WmsgNu=t)	Check to display the type of user messages.
Display informal messages (-WmsgNu=e)	Check to display the informal messages (e.g., memory model, floating point format).
Disable messages about processing statistics (-WmsgNu=d)	Check to disable the information about statistics, e.g., code size, RAM/ROM usage, and so on provided at the end of the assembly.
Disable messages about generated files (-WmsgNu=c)	Check to disable messages informing about generated files.
Disable messages about reading files (-WmsgNu=b)	Check to disable the messages about reading files e.g., the files used as input.
Disable messages about include files (-WmsgNu=a)	Check to disable messages or information provided by the application included files.

3.5.6.7 RS08 Assembler > General

Use this panel to specify the general assembler behavior.

The following table lists and describes the general assembler options for RS08.

Table 3-123. Tool Settings - Assembler > General Options

Option	Description
MMU Support (-MMU)	Check to inform the compiler that CALL and RTC instructions are available, enabling code banking, and that the current architecture has extended data access capabilities, enabling support for `__linear` data types. This option can be used only when `-Cs08` is enabled.
MCUasm compatibility (-MCUasm)	Check to activate the compatibility mode with the MCUasm Assembler.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.

3.5.7 RS08 Preprocessor

Use this panel to configure the preprocessor settings.

The following table lists and describes the Preprocessor options for RS08.

Table 3-124. Tool Settings - Preprocessor Options

Option	Description
Command	Shows the location of the preprocessor executable file. Default value is: `"${HC08Tools}/crs08"`. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} -Lp ${FLAGS} ${INPUTS}`.

3.5.7.1 RS08 Preprocessor > Preprocessor Settings

Use this panel to configure the preprocessor settings of the preprocessor.

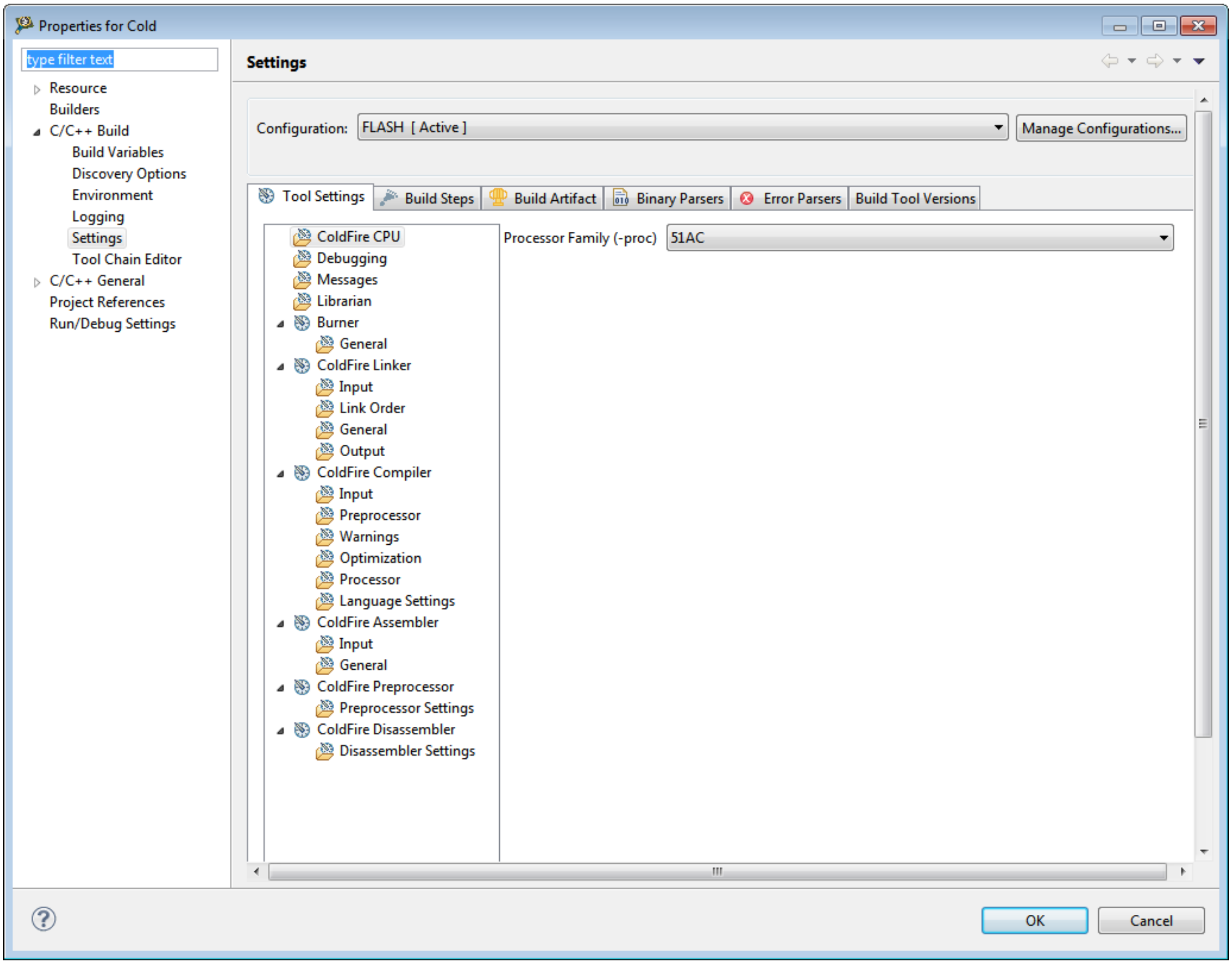
The following table lists and describes the Preprocessor Settings options for RS08.

Table 3-125. Tool Settings - Preprocessor Options

Option	Description
Turn on all preprocessor configuration	Use this option to enable the default preprocessor configuration.
Emit whitespaces (-LpCfg=s)	Use this option to reconstruct spaces.
Handle single quote (`) as normal token (-LpCfg=q)	Use this option to handle single quote (`) as normal token.
Do not concatenate strings (-LpCfg=n)	Use this option to avoid string concatenation.
Emit #line directive (-LpCfg=l)	Use this option to emit #line directives in preprocessor output.
Do not emit file names (-LpCfg=m)	Do not emit file names.
Emit file names with path (-LpCfg=f)	Use this option to emit file names with path.
Emit empty lines (-LpCfg=e)	Use this option to emit empty lines.
Do not emit line comments (-LpCfg=c)	Do not emit line comments
Stop after preprocessor (-LpX)	Without this option, the compiler always translates the preprocessor output as C code. To do only preprocessing, use this option together with the -Lp option. No object file is generated.

3.6 Build Properties for ColdFire

The **Properties for** <*project*> window shows the corresponding build properties for a ColdFire project.



Configuration: FLASH [Active]

Processor Family (-proc) 51AC

Figure 3-21. Build Properties - ColdFire Debug

The following table lists the build properties specific to developing software for ColdFire Debug. The properties that you specify in these panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> window.

Table 3-126. Build Properties for ColdFire Debug

Build Tool	Build Properties Panels
ColdFire CPU	ColdFire CPU
Debugging	Debugging
Messages	Messages

Table continues on the next page...

Table 3-126. Build Properties for ColdFire Debug (continued)

Build Tool	Build Properties Panels
Librarian	Librarian
Burner	Burner > General
ColdFire Linker	ColdFire Linker > Input
	ColdFire Linker > General
	ColdFire Linker > Output
ColdFire Compiler	ColdFire Compiler > Input
	ColdFire Compiler > Preprocessor
	ColdFire Compiler > Warnings
	ColdFire Compiler > Optimization
	ColdFire Compiler > Processor
	ColdFire Compiler > Language Settings
ColdFire Assembler	ColdFire Assembler > Input
	ColdFire Assembler > General
ColdFire Preprocessor	ColdFire Preprocessor > Preprocessor Settings
ColdFire Disassembler	ColdFire Disassembler > Disassembler Settings

3.6.1 ColdFire CPU

Use this panel to specify the CPU type, and the memory model that the architecture uses. The build tools (compiler, linker, and assembler) then use the properties set in this panel to generate CPU-specific code.

The following table lists and describes the ColdFire CPU options.

Table 3-127. Tool Settings - ColdFire CPU Options

Option	Description
Processor Family (-proc)	Lists the processor families supported by the ColdFire compiler. When you select a processor from this list, the compiler generates code that makes use of any of its hardware features or special instructions. For more detailed information on the features of each processor, refer to its reference manual document.

3.6.2 Debugging

Use this panel to specify the whether to generate symbolic information for debugging the build target .

The following table lists and describes the debugging options.

Table 3-128. Tool Settings - Debugging Options

Option	Description
Generate Symbolic Info	Specify whether to generate symbolic information for debugging: • **Off -** Select if you do not want to generate symbolic information for debugging the build target. • **On -** Select to generate symbolic information for debugging the build target. • **Store Full Path Names -** Select to generate symbolic information and store full path names for debugging the build target.

3.6.3 Messages

Use this panel to specify the whether to generate symbolic information for debugging the build target.

The following table lists and describes the message options.

Table 3-129. Tool Settings - Messages Options

Option	Description
Message Style	List options to select message style. • **GCC(default)** - Uses the message style of the Gnu Compiler Collection tools • **MPW** - Uses the Macintosh Programmer's Workshop (MPWï¿½) message style • **Standard** - Uses the standard message style • **IDE** - Uses context-free machine parseable message style • **Enterprise-IDE -** Uses CodeWarrior's Integrated Development Environment (IDE) message style. • **Parseable** - Uses parseable message style.
Maximum Number of Errors	Specify the number of errors allowed until the application stops processing.
Maximum Number of Warnings	Specify the maximum number of warnings.

3.6.4 Librarian

Use this panel to select whether the linker will identify standard libraries.

The following table lists and describes the librarian options.

Table 3-130. Tool Settings - Librarian Options

Option	Description
Enable automatic library configurations	Select to let the compiler identify standard libraries.
Model	Select a standard complying or EWL model from the drop-down list. EWL lets you precisely define the I/O operations. EWL drastically reduces the size of executables as you explicitly select the appropriate I/O behavior. Options are: e wl, c9x, ewl_c++, and c9x_c++.
Print formats	Select the print formats from the drop-down list. The available options are: int, int_FP, int_LL, and int_LL_FP.
Scan formats	Select the scan formats from the drop-down list. The available options are: int, int_FP, int_LL, and int_LL_FP.
IO Mode	Select the input-output mode from the drop-down list. The available options are: raw and buffered.

3.6.5 Burner

Use the Burner for ColdFire Preference Panel to map *.bbl (batch burner language) files to the Burner Plug-In. When the project folder contains a *.bbl file, *.bbl file processing during the post-link phase uses the settings in the Burner preference panel.

The following table lists and describes the burner options for ColdFire.

Table 3-131. Tool Settings - Burner Options

Option	Description
Command	Shows the location of the burner executable file. Default value is: "${HC08Tools}/burner.exe". You can specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the burner will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is ${COMMAND} ${FLAGS} ${INPUTS}.

3.6.5.1 Burner > General

Use this panel to specify other flags for the ColdFire Burner to use.

The following table lists and describes the general options for ColdFire burner.

Table 3-132. Tool Settings - Burner > General Options

Option	Description
Other flags	Specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.

3.6.6 ColdFire Linker

Use this panel to specify ColdFire linker behavior. You can specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the input, general, and output settings.

The following table lists and describes the linker options for ColdFire.

Table 3-133. Tool Settings - ColdFire Linker Options

Option	Description
Command	Shows the location of the linker executable file. Default value is: `"${CF_ToolsDir}/mwldmcf"`. You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the ColdFire linker will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.

3.6.6.1 ColdFire Linker > Input

Use this panel to specify files the ColdFire Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for ColdFire.

Table 3-134. Tool Settings - ColdFire Linker > Input Options

Option	Description
No Standard Library (-nostdlib)	Select if there is no standard library attached.
Link Command File (.lcf)	Consists of three kinds of segments, which must be in this order: • A memory segment, which begins with the MEMORY{} directive • Optional closure segments, which begin with the FORCE_ACTIVE{}, KEEP_SECTION{}, or REF_INCLUDE{} directives • A sections segment, which begins with the SECTIONS{} directive
Entry Point	Specifies the program starting point: the first function the debugger uses upon program start; default: __start. This default function is in file ColdFire__startup.c. It sets up the ColdFire EABI environment before code execution. Its final task is calling main().
Library Search Paths (-L +path)	Specifies the search pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click Workspace or File system, then use the subsequent dialog box to browse to the correct location.
Library Files ?(-l +file)	Specifies the pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click Workspace or File system, then use the subsequent dialog box to browse to the correct location.
Force Active Symbols	Disables deadstripping for particular symbols, enter the symbol names in the Force Active Symbols text box of the ColdFire Linker Panel.

3.6.6.2 ColdFire Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-135. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.6.6.3 ColdFire Linker > General

Use this panel to specify the general linker behavior.

The following table lists and describes the general linker options for ColdFire.

Table 3-136. Tool Settings - ColdFire Linker > General Options

Option	Description
O ther Flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.6.6.4 ColdFire Linker > Output

Use this panel to specify the output settings for the ColdFire linker.

The following table lists and describes the output settings for ColdFire linker.

Table 3-137. Tool Settings - ColdFire Linker > Output Options

Option	Description
Output Type	Select application as Application (default), Static Library, or Shared Library.
Generate Link Map	Check to generate link map.
Generate Link Map - List Unused Objects	Check to generate link map and list unused objects; appears grayed out if the **Generate Link Map** checkbox is not checked.
Generate Link Map -Show Transitive Closure	Check to generate link map and show transitive closure; appears grayed out if the **Generate Link Map** checkbox is not checked.
Generate Link Map -Always Keep Map	Check to generate link map and always keep the map; appears grayed out if the **Generate Link Map** checkbox is not checked.
Generate Link Map - Generate S-Record File	Check to generate link map and generate a S-record file.
Max S-Record Length	Specify the maximum length for S-record; appears grayed out if the **Generate S-Record File** checkbox is not checked. The default value is 252.
EOL Character	Specify the end-of-line character; appears grayed out if the **Generate S-Record File** checkbox is not checked. The default value is DOS.
Generate Listing File	Check to generate a listing file named lstfil.lst.
Generate Elf Symbol Table	Check to generate an ELF symbol table.

Table continues on the next page...

Table 3-137. Tool Settings - ColdFire Linker > Output Options (continued)

Option	Description
Generate Binary Image	Check to generate a binary image.
Max Bin Record	Specify the maximum value for bin record; appears grayed out if the **Generate Binary Image** checkbox is not checked. The default value is 252.
Generate Raw-Binary Image	Check to generate a raw-binary image.
Max Raw-Binary Gap	Specify the maximum value for raw binary gap; appears grayed out if the **Generate Raw-Binary Image** checkbox is not checked. The default value is 0x10000.
Generate Warning Messages	Select whether you want to generate warning messages, warn superseded definitions, or treat warnings as errors.

3.6.7 ColdFire Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the ColdFire Compiler tree control includes the general and the file search path settings.

The following table lists and describes the compiler options for ColdFire.

Table 3-138. Tool Settings - Compiler Options

Option	Description
Command	Shows the location of the compiler executable file. Default value is: `"${CF_ToolsDir}/mwccmcf"`. You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the ColdFire compiler will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.

3.6.7.1 ColdFire Compiler > Input

Use this panel to specify additional files the ColdFire Compiler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for ColdFire compiler.

Table 3-139. Tool Settings - ColdFire Compiler > Input Options

Option	Description
Compile only, Do Not Link (-c)	Check if you want to compile only and do not want to link the file.
Do not use MWCIncludes variable (-nostdinc)	Check if you do not want to use MWCIncludes variable.
Always Search User Paths (-nosyspath)	Check if you want to always search user paths.
Source encoding	Lets you specify the default encoding of source files. The compiler recognizes Multibyte and Unicode source text. To replicate the obsolete option Multi-Byte Aware, set this option to System or Autodetect. Additionally, options that affect the preprocess request appear in this panel. The options available are as follows: • ASCII • Autodetect • UTF-8 • System • Shift-JIS • EUC-JP • ISO-2022-JP
User Path (-i)	Lists the available user paths.
User Recursive Path (-ir)	Appends a recursive access path to the current #include list. This command is global. **Syntax**`-ir pathpath` The recursive access path to append.
System Path (-I- -I)	Lists the available system paths.
System Recursive Path (-I- -ir)	Lists the available system paths recursively.

The following table lists and describes the toolbar buttons that help work with the user and system search paths.

Table 3-140. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the search path.
✖	Delete - Click to delete the selected search path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected search path.
⇧	Move up - Click to move the selected search path one position higher in the list
⇩	Move down - Click to move the selected search path one position lower in the list

The following table shows the Add directory path dialog box.

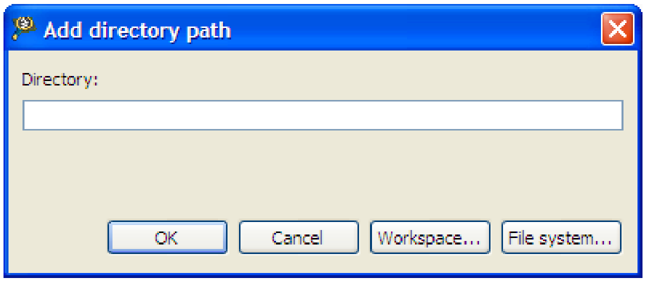
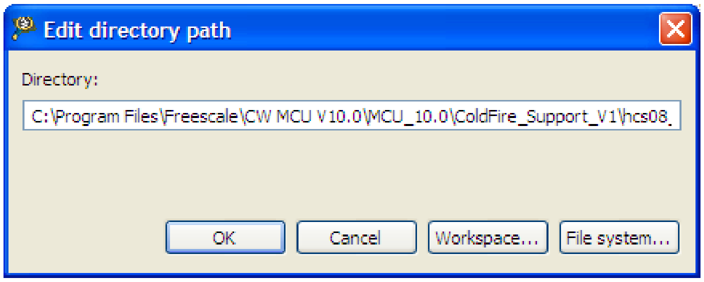


Figure 3-22. Add directory path Dialog Box

The following table shows the Edit directory path dialog box.



Directory: C:\Program Files\Freescale\CW MCU V10.0\MCU_10.0\ColdFire_Support_V1\hcs08_

Figure 3-23. Edit directory path Dialog Box

The buttons in the **Add directory path and Edit directory path** dialog boxes help work with the object file search paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the object file search path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.

3.6.7.2 ColdFire Compiler > Preprocessor

Use this panel to specify ColdFire preprocessor behavior.

The following table lists and describes the preprocessor options for ColdFire Compiler.

Table 3-141. Tool Settings - ColdFire Compiler > Preprocessor

Option	Description
Prefix File (-prefix)	Specifies a file automatically included in all project assembly files.

Table continues on the next page...

Table 3-141. Tool Settings - ColdFire Compiler > Preprocessor (continued)

Option	Description
Defined Macros (-D)	Lists the defined command-line macros.
Undefined Macros (-U)	Lists the undefined command-line macros.

3.6.7.3 ColdFire Compiler > Warnings

Use this panel to control how the ColdFire compiler formats the listing file, as well as error and warning messages.

The following table lists and describes the warnings options for ColdFire compiler.

Table 3-142. Tool Settings - ColdFire Compiler > Warnings Options

Option	Description
Treat All Warnings As Errors	Check to treat all warnings as errors. The compiler will stop if it generates a warning message.
Illegal Pragmas	Check to notify the presence of illegal pragmas.
Possible Errors	Check to suggest possible errors.
Extended Error Checking	Check if you want to do an extended error checking.
Hidden virtual functions	Check to generate a warning message if you declare a non-virtual member function that prevents a virtual function, that was defined in a superclass, from being called and is equivalent to pragma `warn_hidevirtual` and the command-line option `-warnings hidevirtual.`
Implicit Arithmentic Conversions	Check to warn of implicit arithmetic conversions.
Implicit Integer to Float Conversions	Check to warn of implicit conversion of an integer variable to floating-point type.
Implicit Float to Integer Conversions	Check to warn of implicit conversions of a floating-point variable to integer type.
Implicit Signed/Unsigned Conversion	Check to enable warning of implicit conversions between signed and unsigned variables.
Pointer/Integral Conversions	Check to enable warnings of conversions between pointer and integers.
Unused Arguments	Check to warn of unused arguments in a function.
Unused Variables	Check to warn of unused variables in the code.
Unused Result From Non-Void-Returning Function	Check to warn of unused result from non-void-returning functions.
Missing `return' Statement	Check to warn of when a function lacks a return statement.
Expression Has No Side Effect	Check to issue a warning message if a source statement does not change the program's state. This is equivalent to the pragma `warn_no_side_effect`, and the command-line option `-warnings unusedexpr.`

Table continues on the next page...

Table 3-142. Tool Settings - ColdFire Compiler > Warnings Options (continued)

Option	Description
Extra Commas	Check to issue a warning message if a list in an enumeration terminates with a comma. The compiler ignores terminating commas in enumerations when compiling source code that conforms to the ISO/IEC 9899-1999 ("C99") standard and is equivalent to pragma `warn_extracomma` and the command-line `option -warnings extracomma.`
Empty Declarations	Check to warn of empty declarations.
Inconsistent `class' / `struct' Usage	Check to warn of inconsistent usage of class or struct.
Include File Capitalization	Check to issue a warning message if the name of the file specified in a #include "file" directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps` and the command-line option `-warnings filecaps.`
Check System Includes	Check to issue a warning message if the name of the file specified in a #include <file> directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps_system` and the command-line option `-warnings sysfilecaps.`
Pad Bytes Added	Check to issue a warning message when the compiler adjusts the alignment of components in a data structure and is equivalent to pragma `warn_padding` and the command-line option `-warnings padding.`
Undefined Macro in #if	Check to issues a warning message if an undefined macro appears in #if and #elif directives and is equivalent to pragma `warn_undefmacro` and the command-line option `-warnings undefmacro.`
Non-Inlined Functions	Check to issue a warning message if a call to a function defined with the inline, __inline__, or __inline keywords could not be replaced with the function body and is equivalent to pragma `warn_notinlined` and the command-line option `-warnings notinlined.`
Token not formed by ## operator	Check to enable warnings for the illegal uses of the preprocessor's token concatenation operator (##). It is equivalent to the pragma `warn_illtokenpasting on`.

3.6.7.4 ColdFire Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and defines each option of the **Optimization** panel.

Table 3-143. Tool Settings - ColdFire Compiler > Optimization Options

Option	Description
Optimization Level (-opt)	Specify the optimizations that you want the compiler to apply to the generated object code: • 0-Disable optimizations. This setting is equivalent to specifying the `-O0` command-line option. The compiler generates unoptimized, linear assembly-language code. • 1-The compiler performs all target-independent (that is, non-parallelized) optimizations, such as function inlining. This setting is equivalent to specifying the `-O1` command-line option. The compiler omits all target-specific optimizations and generates linear assembly-language code. • 2-The compiler performs all optimizations (both target-independent and target-specific). This setting is equivalent to specifying the `-O2` command-line option. The compiler outputs optimized, non-linear, parallelized assembly-language code. • 3-The compiler performs all the level 2 optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the `-O3` command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level 2 optimizations.
Speed Vs Size	Use to specify an Optimization Level greater than 0. • Speed-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a faster execution speed, as opposed to a smaller executable code size. • Size-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a smaller executable code size, as opposed to a faster execution speed. This setting is equivalent to specifying the `-Os` command-line option.
IPA	Specifies the Interprocedural Analysis (IPA) policy. • **Off** - No interprocedural analysis, but still performs function-level optimization. Equivalent to the "no deferred inlining" compilation policy of older compilers. • **File** - Completely parse each translation unit before generating any code or data. Equivalent to the "deferred inlining" option of older compilers. Also performs an early dead code and dead data analysis in this mode. Objects with unreferenced internal linkages will be dead-stripped in the compiler rather than in the linker.
Inlining	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. Using the -Oi=c0 option switches off inlining. Functions marked with the #pragma INLINE are still inlined. To disable inlining, use the -Oi=OFF option.

Table continues on the next page...

Table 3-143. Tool Settings - ColdFire Compiler > Optimization Options (continued)

Option	Description
Bottom-up Inlining	Check to control the bottom-up function inlining method. When active, the compiler inlines function code starting with the last function in the chain of functions calls, to the first one.

3.6.7.5 ColdFire Compiler > Processor

Use this panel to specify processor behavior. You can specify the file paths and define macros.

The following table lists and defines each option of the Processor panel.

Table 3-144. Tool Settings - ColdFire Compiler > Processor Options

Option	Description
Struct Align (-align)	Specifies record and structure alignment in memory: • **Byte** - Aligns all fields on 1 byte boundaries • **68k (word)** - Aligns all fields on word boundaries • **coldfire (long)** - Aligns all fields on long word boundaries • **Default -** Coldfire (long). This panel element corresponds to the options align pragma. **Note** : When you compile and link, ensure that alignment is the same for all files and libraries.
Code Model	Specifies access addressing for data and instructions in the object code: • **Smart** - Relative (16-bit) for function calls in the same segment; otherwise absolute (32-bit) • **Far (32 bit)** - Absolute for all function calls • **Near (16 bit)** - Relative for all function calls • **Near Relative (pc16)** - Generates a 16-bits relative references to code.
Data Model	Specifies global-data storage and reference: • **Far (32 bit)** - Storage in far data space; available memory is the only size limit. • **Near (16 bit)** - Storage in near data space; size limit is 64K. • **Default** - Far (32 bit). This panel element corresponds the far_data pragma
Floating Point	Specifies handling method for floating point operations: • **Software** - C runtime library code emulates floating-point operations. • **Hardware** - Processor hardware performs floating point operations; only appropriate for processors that have floating-point units. • **None**

Table continues on the next page...

Table 3-144. Tool Settings - ColdFire Compiler > Processor Options (continued)

Option	Description
	Default: Software For software selection, your project must include the appropriate FP_ColdFire C runtime library file. Grayed out if your target processor lacks an FPU.
A6 Stack Frame (-a6)	Clear to disable call-stack tracing; generates faster and smaller code. By default, the option is checked.
Pool Sections (-pool)	Check to collect all string constants into a single data object so your program needs one data section for all of them.
Generate Code for Profiling (-profile)	Check to enable the processor generate code for use with a profiling tool. Checking this box corresponds to using the command-line option `-profile`. Clearing this checkbox is equivalent to using the command-line option `-noprofile`
Position-Independent Code (-pic)	Check to generate position independent code (PIC) that is non relocatable.
Position-Independent Data (-pid)	Check to generate non-relocatable position-independent data (PID). PID is available with 16- and 32-bit addressing.
Register Coloring (-coloring)	Clear to enable the Compiler force all local variables to be stack-based except for compiler generated temporaries.
Instruction Scheduling (-scheduling)	Clear to prevent from scheduling instructions.
Peephole (-peephole)	Clear to prevent the compiler from compiling long instruction sequences into compact ones. By default, the option is checked. When on (default setting) it does not affect debugging unless the resulting instruction is a memory-to-memory operation which might make a variable used as temporary disappear.
Use .sdata.sbiss for (byte in integer between -1.32K)	The options are: • All data - Select this option button to store all data items in the small data address space • All data smaller than - Select this option button to specify the maximum size for items stored in the small data address space; enter the maximum size in the text box. Using the small data area speeds data access, but has ramifications for the hardware memory map. The default settings specify not using the small data area. By default, all data smaller than is checked.

3.6.7.6 ColdFire Compiler > Language Settings

Use this panel direct the ColdFire compiler to apply specific processing modes to the language source code. You can compile source files with just one collection at a time. To compile source files with multiple collections, you must compile the source code sequentially. After each compile iteration change the collection of settings that the ColdFire compiler uses.

The following table lists and defines each option of the Language Settings panel.

Table 3-145. Tool Settings - ColdFire Compiler > Language Settings Options

Option	Description
Require Prototypes (-requireprotos)	Check to enforce the requirement of function prototypes. the compiler generates an error message if you define a previously referenced function that does not have a prototype. If you define the function before it is referenced but do not give it a prototype, this setting causes the compiler to issue a warning message.
Enable C++ \`bool' type, \`true' and \`false' Constants (-bool)	Check to enable the C++ compiler recognize the bool type and its true and false values specified in the ISO/IEC 14882-1998 C++ standard; is equivalent to pragma `bool` and the command-line option `-bool`.
ISO C++ Template Parser (-iso_templates)	Check to follow the ISO/IEC 14882-1998 standard for C++ to translate templates, enforcing more careful use of the typename and template keywords. The compiler also follows stricter rules for resolving names during declaration and instantiation and is equivalent to pragma `parse_func_templ` and the command-line `option -iso_templates`.
Use Instance Manager (-inst)	Check to reduce compile time by generating any instance of a C++ template (or non-inlined inline) function only once.
Force C++ Compilation (-lang c99)	Check to translates all C source files as C++ source code and is equivalent to pragma `cplusplus` and the command-line option `-lang c++`.
Enable GCC extensions (-gcc)	Check to recognize language features of the GNU Compiler Collection (GCC) C compiler that are supported by CodeWarrior compilers; is equivalent to pragma `gcc_extensions` and the command-line option `-gcc_extensions`.
Enable C99 Extensions (-lang c99)	Check to recognize ISO/IEC 9899-1999 ("C99") language features; is equivalent to pragma `c99` and the command-line option `- dialect c99`.
Enable C++ Exceptions (-Cpp_Exceptions)	Check to generate executable code for C++ exceptions; is equivalent to pragma `exceptions` and the command-line option `- cpp_exceptions`.
Enable RTTI (-RTTI)	Check to allow the use of the C++ runtime type information (RTTI) capabilities, including the `dynamic_cast` and `typeid` operators; is equivalent to pragma RTTI and the command-line option -RTTI.
Enable wchar_tSupport	Check to enable C++ compiler recognize the wchar_t data type specified in the ISO/IEC 14882-1998 C++ standard; is equivalent to pragma `wchar_type` and the command-line option `-wchar_t`.
ANSI Strict	Check to enable C compiler operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ISO specification to all input files. This setting is equivalent to specifying the `- ansi` command-line option. The compiler issues a warning for each ANSI/ISO extension it finds.

Table continues on the next page...

Table 3-145. Tool Settings - ColdFire Compiler > Language Settings Options (continued)

Option	Description
ANSI Keywords Only	Check to generate an error message for all non-standard keywords (ISO/IEC 9899-1990 C, ï¿½6.4.1). If you must write source code that strictly adheres to the ISO standard, enable this setting; is equivalent to pragma `only_std_keywords` and the command-line option `-stdkeywords`.
Expand Trigraphs	Check to recognize trigraph sequences (ISO/IEC 9899-1990 C, ï¿½5.2.1.1); is equivalent to **pragma** `trigraphs` **and the command-line option** `-trigraphs`.
Legacy for-scoping	Check to generate an error message when the compiler encounters a variable scope usage that the ISO/IEC 14882-1998 C++ standard disallows, but is allowed in the C++ language specified in The Annotated C++ Reference Manual; is equivalent to pragma `ARM_scoping` and the command-line option `-for_scoping`.
Enum Always Int	Check to use signed integers to represent enumerated constants and is equivalent to pragma `enumsalwaysint` and the command-line option `-enum`.
Use Unsigned Chars	Check to treat char declarations as unsigned char declarations and is equivalent to pragma `unsigned_char` and the command-line option `-char unsigned`.
Pool Strings	**Check to** collect all string constants into a single data section in the object code it generates and is equivalent to pragma `pool_strings` and the command-line `option -strings pool`.
Reuse Strings	**Check to** store only one copy of identical string literals and is equivalent to opposite of the pragma `dont_reuse_strings` and the command-line `option -string reuse`.
Other flags	Specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI. **Note** : To enable CodeWarrior MCU V10.x to generate .lst file for each source file in ColdFire, you need to specify -S in the **Other Flags** option.

3.6.8 ColdFire Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler. Additionally, the Assembler tree control includes the general and include file search path settings.

The following table lists and defines each option of the ColdFire Assembler panel.

Table 3-146. Tool Settings - ColdFire Assembler Options

Option	Description
Command	Shows the location of the assembler executable file. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.
Command line pattern	

3.6.8.1 ColdFire Assembler > Input

Use this panel to specify additional files the ColdFire Assembler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for ColdFire assembler.

Table 3-147. Tool Settings - ColdFire Assembler > Input Options

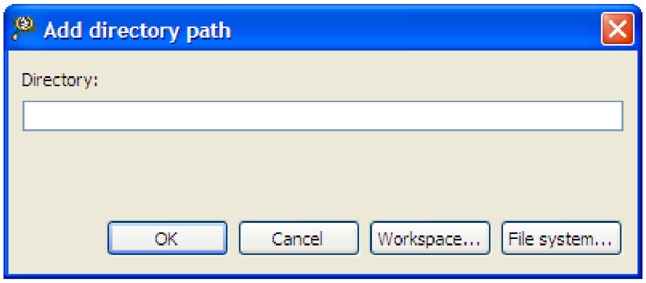
Option	Description
Prefix File	Specifies a file automatically included in all project assembly files.
Always Search User Paths (-nosyspath)	CodeWarrior searches only the system path when looking for a file included with #include <...>. To have it searches user path as well, check this box. Note: #include "..." will always search both sets of paths.
User Path (-i)	Lists the available user paths.
User Recursive Path (-ir)	Appends a recursive access path to the current #include list. This command is global. **Syntax**`-ir pathpath` The recursive access path to append.
System Path (-I- -I)	Lists the available system paths.
System Recursive Path (-I- -ir)	Appends a system recursive access path to the current #include list. This command is global.

The following table lists and describes the toolbar buttons that help work with the user and system search paths.

Table 3-148. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the search path.
✖	Delete - Click to delete the selected search path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected search path.
⇧	Move up - Click to move the selected search path one position higher in the list
⇩	Move down - Click to move the selected search path one position lower in the list

The following figure shows the Add directory path dialog box.

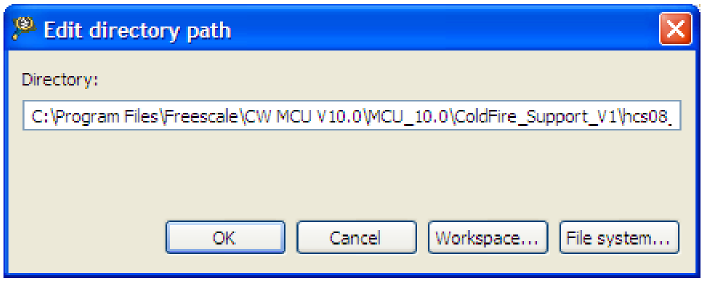


Add directory path

Directory:

Figure 3-24. Add directory path Dialog Box

The following table shows the Edit directory path dialog box.



Edit directory path

Directory: C:\Program Files\Freescale\CW MCU V10.0\MCU_10.0\ColdFire_Support_V1\hcs08_

Figure 3-25. Edit directory path Dialog Box

The buttons in the **Add directory path and Edit directory path** dialog boxes help work with the object file search paths.

- **OK -** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

- Workspace **-** Click to display the **Folder Selection** dialog box and specify the object file search path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.

3.6.8.2 ColdFire Assembler > General

Use this panel to specify the general assembler behavior.

The following table lists and describes the general assembler options for ColdFire.

Table 3-149. Tool Settings - Assembler > General Options

Option	Description
Label Must End With `:'	Clear if system does not require labels to end with colons. By default, the option is checked.
Directives Begin With `.'	Clear if the system does not require directives to start with periods. By default, the option is checked.
Case Sensitive Identifier	Clear to instruct the assembler to ignore case in identifiers. By default, the option is checked.
Allow Space In Operand Field	Clear to restrict the assembler from adding spaces in operand fields. By default, the option is checked.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.

3.6.9 ColdFire Preprocessor

Use this panel to specify preprocessor behavior and define macros.

The following table lists and describes the preprocessor options for ColdFire.

Table 3-150. Tool Settings - ColdFire Preprocessor Options

Option	Description
Command	Shows the location of the preprocessor executable file. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the preprocessor will be called with

Table continues on the next page...

Table 3-150. Tool Settings - ColdFire Preprocessor Options (continued)

Option	Description
Expert Settings	Shows the command line pattern; default `${COMMAND} ${FLAGS} ${INPUTS}`
Command line pattern	

3.6.9.1 ColdFire Preprocessor > Preprocessor Settings

Use this panel to specify preprocessor behavior.

The following table lists and describes the preprocessor options for ColdFire.

Table 3-151. Tool Settings - ColdFire Compiler > Preprocessor Options

Option	Description
Emit file change (-ppopt break)	Check to notify file changes (or #line changes) appear in the output.
Emit #pragmas (-ppopt pragma)	Check to show pragma directives in the preprocessor output. Essential for producing reproducible test cases for bug reports.
Show full path (-ppopt full)	Check to display file changes in comments (as before) or in #line directives.
Keep comment (-ppopt comment)	Check to display comments in the preprocessor output.
Use #include line (-ppopt line)	Check to display file changes in comments (as before) or in #line directives.
Keep whitespace (-ppopt nospace)	Check to copy whitespaces in preprocessor output. This is useful for keeping the starting column aligned with the original source, though the compiler attempts to preserve space within the line. This does not apply when macros are expanded.

3.6.10 ColdFire Disassembler

Use this panel to specify the command, options, and expert settings for ColdFire Disassembler.

The following table lists and describes the ColdFire disassembler options.

Table 3-152. Tool Settings - Linker Options

Option	Description
Command	Shows the location of the linker executable file. You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with
Expert Settings	Shows the command line pattern; default is `${COMMAND} -dis ${FLAGS} ${INPUTS}`
Command line pattern	

3.6.10.1 ColdFire Disassembler > Disassembler Settings

Use this panel to control how the disassembler formats the listing file, as well as error and warning messages. You can specify verbosity of messages, whether to show headers, core modules, extended mnemonics, addresses, object or source code, ldata modules, exception tables, and debug information.

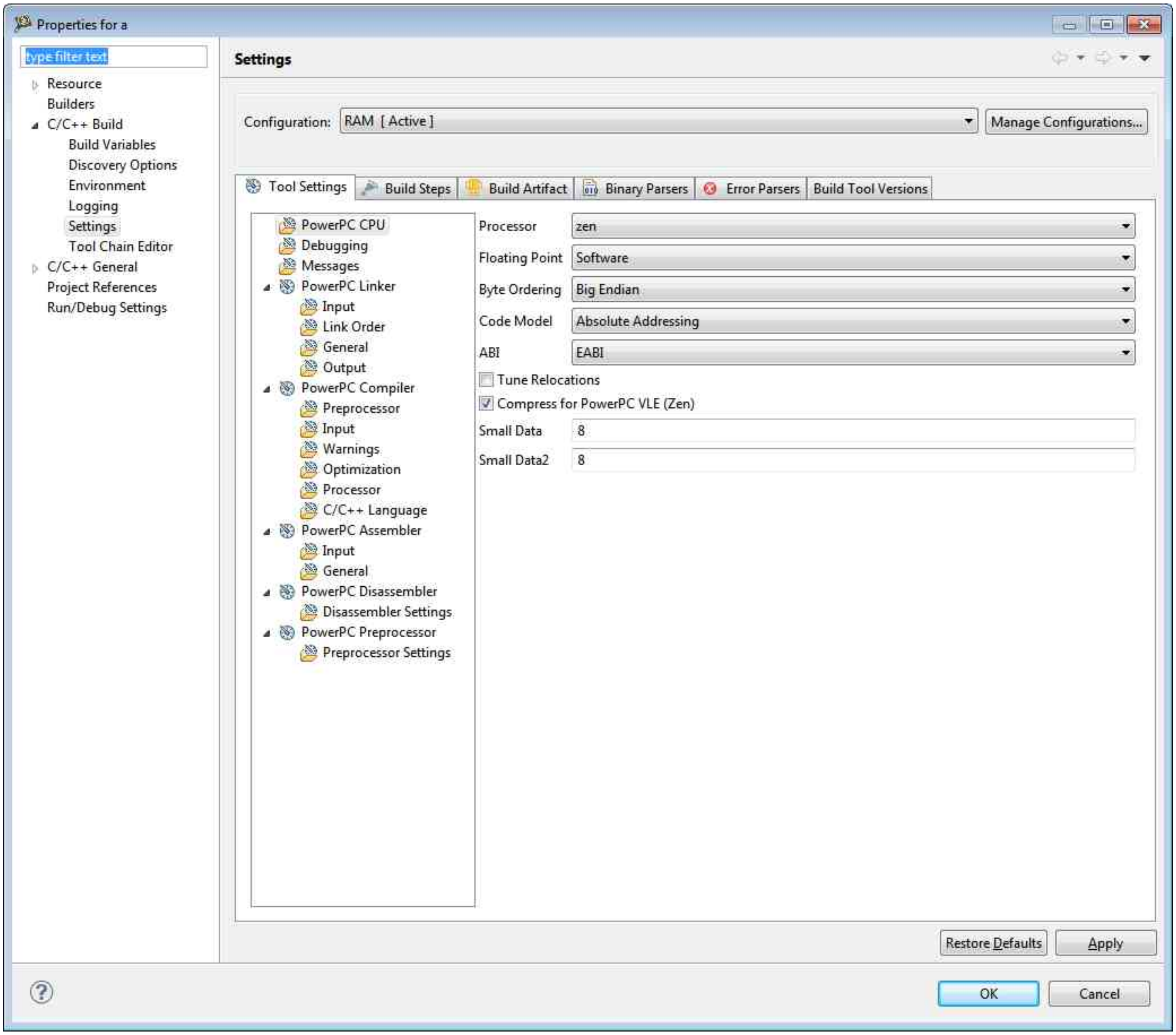
The following table lists and describes the ColdFire disassembler settings.

Table 3-153. Tool Settings - ColdFire Disassembler Options

Option	Description
Show Headers	Check to display headers in the listing file; disassembler writes listing headers, titles, and subtitles to the listing file
Show Symbol and String Tables	Check to display symbol and string tables directives to the listing file
Verbose Info	Check to shows each command line that it passes to the shell, along with all progress, error, warning, and informational messages that the tools emit
Show Relocations	Check to have the disassembler show information about relocated symbols. Clear to prevent the disassembler from showing information about relocated symbols.
Show Core Modules	Check to show core modules in the listing file
Show Extended Mnemonics	Check to show the extended mnemonics in the listing file
Show Addresses and Object Code	Check to show the addresses and object code in the listing file
Show Source Code	Check to show the source code in the listing file
Show Comments	Check to show the comments in the listing file
Show Data Modules	Check to show the data modules in the listing file
Disassemble Exception Tables	Check to disassemble exception tables in the listing file
Show Debug Info	Check to generate symbolic information for debugging the build target

3.7 Build Properties for Qorivva

The **Properties for** <*project*> dialog box shows the corresponding build properties for a Qorivva project.



Configuration: RAM [Active]

Processor: zen

Floating Point: Software

Byte Ordering: Big Endian

Code Model: Absolute Addressing

ABI: EABI

□ Tune Relocations

☑ Compress for PowerPC VLE (Zen)

Small Data: 8

Small Data2: 8

Figure 3-26. Build Properties - Qorivva

The following table lists the build properties specific to developing software for Qorivva. The properties that you specify in these panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> dialog box.

Table 3-154. Build Properties for Qorivva

Build Tool	Build Properties Panels
PowerPC CPU	PowerPC CPU
Debugging	Debugging
Messages	Messages
PowerPC Linker	PowerPC Linker > Input
	PowerPC Linker > General
	PowerPC Linker > Output
PowerPC Compiler	PowerPC Compiler > Preprocessor
	PowerPC Compiler > Input
	PowerPC Compiler > Warnings
	PowerPC Compiler > Optimization
	PowerPC Compiler > Processor
	PowerPC Compiler > C/C++ Language
PowerPC Assembler	PowerPC Assembler > Input
	PowerPC Assembler > General
PowerPC Disassembler	PowerPC Disassembler > Disassembler Settings
PowerPC Preprocessor	PowerPC Preprocessor > Preprocessor Settings

3.7.1 PowerPC CPU

Use this panel to specify the CPU type, and the memory model that the architecture uses. The build tools (compiler, linker, and assembler) then use the properties set in this panel to generate CPU-specific code.

The following table lists and describes the PowerPC CPU options.

Table 3-155. Tool Settings - PowerPC CPU

Option	Description
Processor	Lists the processor families supported by the Power Architecture compiler. When you select a processor from this list, the compiler generates code that makes use of any of its hardware features or special instructions. For more detailed information on the features of each processor, refer to its reference manual document.

Table continues on the next page...

Table 3-155. Tool Settings - PowerPC CPU (continued)

Option	Description
Floating Point	Define how the compiler handles floating-point operations it encounters in the source code. • **Software** - Select to have the compiler emulate floating-point operations by calling functions that perform floating-point math. The C runtime library contains the functions the compiler invokes. If you use software floating-point emulation, you must include the appropriate C runtime library in your project. Enabling this option without including the appropriate C runtime library causes link errors. • **Hardware** - Select to have the compiler handle floating-point operations by generating instructions for the hardware floating-point unit. Do not select this option if your target processor does not have a hardware floating-point unit. • **None** - Select to disable floating-point support. • **SPFP** - Select to have the compiler handle single-precision floating-point operations by generating instructions for the e500-EFPU floating point unit, and perform double-precision floating-point operations by calling functions that perform double-precision floating-point math. Do not select this option if your target processor does not have a e500-EFPU floating-point unit. • **SPFP_Only** - Select to have the compiler handle single-precision floating-point operations by generating instructions for the e500-EFPU floating point unit. • **DPFP** - Select to have the compiler handle both single- and double-precision floating-point operations by generating instructions for the e500 DPFP APU (Double-Precision Floating-Point Auxiliary Processing Unit). Do not select this option if your target processor does not have a DPFPunit. **Default** : Software If the selected processor does not handle a floating-point exception, you should select `None` or `Software` floating-point support.
Byte Ordering	Enables you to select big-endian or little-endian byte ordering. • **Big Endian** - Select to generate object code and links an executable image that uses big-endian byte ordering. This is the default setting for the compiler and linker. If you choose big-endian byte ordering, within a given multi-byte numeric representation, the most significant byte has the lowest address; the word is stored big-end-first. • **Small Endian** - Select to generates object code and links an executable image that uses little-endian byte ordering. If you choose little endian byte ordering, within

Table continues on the next page...

Table 3-155. Tool Settings - PowerPC CPU (continued)

Option	Description
	a given multi-byte numeric representation, bytes at lower addresses have lower significance; the word is stored little-end-first.
Code Model	Enables you to select the addressing mode for the binary generated by the current build target. • **Absolute Addressing** - Select to instruct the build tools to generate a non-relocatable binary. • **SDA Based PIC/PID Addressing** - Select to instruct the build tools to generate a relocatable binary that uses position independent-code (PIC)/position-independent-data (PID) addressing. The resulting binary can be loaded at any address.
ABI	Enables you to select the Application Binary Interface (ABI) the compiler and assembler use for function calls and structure layout. • **EABI** - Converts a 14-bit branch relocation to a 24-bit branch relocation only if the 14-bit relocation cannot reach the calling site from the original relocation. • **System V ABI** - Changes the absolute addressed references of data from code to use a small data register (such as r13) instead of r0; absolute code is changed to code references to use the PC relative relocations. • **SuSE** - Use the SuSEï¿½ Linux ABI with GNU extensions. • **YellowDog** - Use the YellowDog™ Linux ABI with GNU extensions. • **SDA PIC/PID** - Use position-independent addressing executable code and data.
Tune Relocations	Pertains to object relocation and is available for just the above mentioned application binary interfaces.
Compress for PowerPC VLE (Zen)	Generate VLE instructions. This option sets the processor to Zen.
Small Data	Specify the threshold size (in bytes) for an item to be considered small data by the linker. The linker stores small data items in the Small Data address space. Data in the Small Data address space can be accessed more quickly than data in the normal address space.
Small Data2	Specify the threshold size (in bytes) for an item to be considered small data by the linker. The linker stores read-only small data items in the Small Data2 address space. Constant data in the Small Data2 address space can be accessed more quickly than data in the normal address space.

3.7.2 Debugging

Use this panel to specify the whether to generate symbolic information for debugging the build target.

The following table lists and describes the debugging options.

Table 3-156. Tool Settings - Debugging

Option	Description
Generate DWARF Information	Select the version of the Debug With Arbitrary Record Format (DWARF) debugging information format the compiler and assembler generates. If in doubt about the DWARF version to use, you can use the default setting of DWARF 2.x. The linker ignores debugging information that is not in the selected format.
Store Full Paths to Source Files	Store absolute paths of source files instead of relative paths.

3.7.3 Messages

Use this panel to specify the whether to generate symbolic information for debugging the build target.

The following table lists and describes the message options.

Table 3-157. Tool Settings - Messages Options

Option	Description
Message Style	List options to select message style. • **GCC** - Uses the message style of the Gnu Compiler Collection tools • **MPW** - Uses the Macintosh Programmer's Workshop (MPWï¿½) message style • **standard** - Uses the standard message style • **IDE** - Uses context-free machine parseable message style • **parseable** - Uses parseable message style. This is default. • **Enterprise-IDE -** Uses CodeWarrior's Integrated Development Environment (IDE) message style.
Maximum Number of Errors	Specify the number of errors allowed until the application stops processing.
Maximum Number of Warnings	Specify the maximum number of warnings.

3.7.4 PowerPC Linker

Use this panel to specify PowerPC linker behavior. You can specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the input, general, and output settings.

The following table lists and describes the linker options for PowerPC.

Table 3-158. Tool Settings - PowerPC Linker

Option	Description
Command	Shows the location of the linker executable file. Default: `${PAToolsDir}/mwldeppc.` You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert settings	Shows the command line pattern. Default: `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`
Command line pattern	

3.7.4.1 PowerPC Linker > Input

Use this panel to specify files the PowerPC Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for PowerPC.

Table 3-159. Tool Settings - PowerPC Linker > Input

Option	Description
No Standard Library	Select if there is no standard library attached.
Link Command File (.lcf)	Consists of three kinds of segments, which must be in this order: • A memory segment, which begins with the MEMORY{} directive • Optional closure segments, which begin with the FORCE_ACTIVE{}, KEEP_SECTION{}, or REF_INCLUDE{} directives • A sections segment, which begins with the SECTIONS{} directive
Code Address	Specifies the location in memory where the executable code resides. The possible addresses depend on your target hardware platform and how the memory is mapped.
Data Address	Enables you to specify the address for global data.
Small Data Address	Enables you to specify the RAM address for the first small data section. This address must not conflict with the target-hardware memory map; target hardware must support this address.

Table continues on the next page...

Table 3-159. Tool Settings - PowerPC Linker > Input (continued)

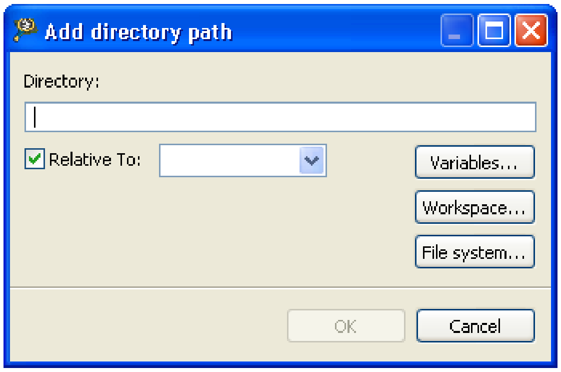
Option	Description
Small Data 2 Address	Enables you to specify the RAM address for the second small data section. This address must not conflict with the target-hardware memory map; target hardware must support this address.
Entry Point	Enables you to specify the program starting point - the function that the linker uses first when you launch the program. This default function (in file __start.c) is bootstrap/ glue code that sets up the EABI environment, then calls function main().
Library Search Paths	Enables you to specify the search pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click Workspace or File system, then use the subsequent dialog box to browse to the correct location.
Library Files	Enables you to specify the pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click Workspace or File system, then use the subsequent dialog box to browse to the correct location.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object file search paths.

Table 3-160. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the Add directory path dialog box.

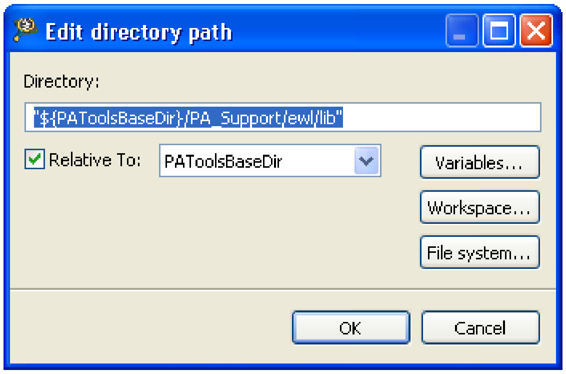


Directory:

☑ Relative To:

Figure 3-27. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



Directory: "${PAToolsBaseDir}/PA_Support/ewl/lib"

☑ Relative To: PAToolsBaseDir

Figure 3-28. Edit directory path Dialog Box

The buttons in the **Add directory path and Edit directory path** dialog boxes help work with the object file search paths.

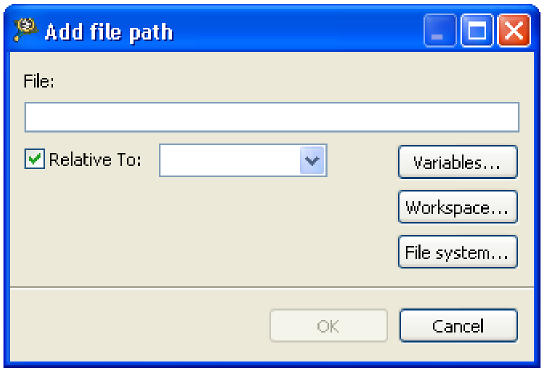
- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object files.

Table 3-161. Libraries Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add file path** dialog box and specify location of the library you want to add.
✖	Delete - Click to delete the selected library path. To confirm deletion, click **Yes** in the **Confirm Delete** dialog box.
✎	Edit - Click to open the **Edit file path** dialog box and update the selected path.
⇧	Move up - Click to move the selected path one position higher in the list.
⇩	Move down - Click to move the selected path one position lower in the list.

The following figure shows the Add file path dialog box.



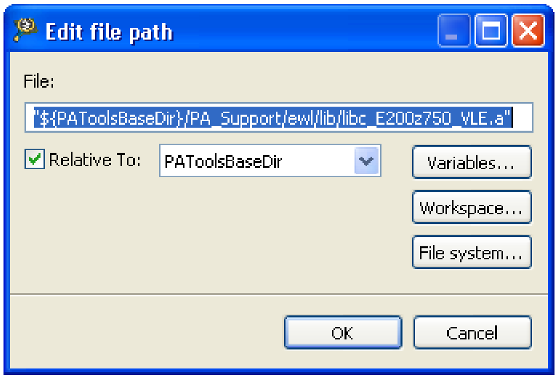
Add file path

File:

☑ Relative To:

Figure 3-29. Tool Settings - Linker > Libraries - Add file path Dialog Box

The following figure shows the Edit file path dialog box.



Edit file path

File: "${PAToolsBaseDir}/PA_Support/ewl/lib/libc_E200z750_VLE.a"

☑ Relative To: PAToolsBaseDir

Figure 3-30. Tool Settings - Linker > Libraries - Edit file path Dialog Box

The buttons in the **Add file path** and **Edit file path** dialog boxes help work with the file paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the file path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **File Selection** dialog box and specify the file path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Open** dialog box and specify the file path. The resulting absolute path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.7.4.2 PowerPC Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-162. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.7.4.3 PowerPC Linker > General

Use this panel to specify the memory model that the architecture uses. The build tools (compiler, linker, and assembler) use the properties that you specify.

The following table lists and describes the memory model options for PowerPC.

Table 3-163. Tool Settings - PowerPC Linker > General

Option	Description
Link Mode	Specifies how much memory the linker uses to write output to the hard disk. • **Normal** - Writes to a 512-byte buffer, then writes the buffer to disk.

Table continues on the next page...

Table 3-163. Tool Settings - PowerPC Linker > General (continued)

Option	Description
	• **Use Less RAM** - Writes output file directly to disk, without using a buffer. • **Use More RAM** - Writes each segment to its own buffer, then flushes all buffers to the disk. Linking requires enough RAM space for all input files and numerous housekeeping structures. Normal is the best choice for most projects; **Use More RAM** is appropriate for small projects.
Code Merging	Controls merging optimization. • **Off** • **Safe Functions** • **All Functions** Checking **Off** deactivates the **Aggressive Merging** checkbox.
Aggressive Merging	Check to implement aggressive merging. This checkbox is active only if **Safe Functions** or **All Functions** is selected in the **Code Merging** drop-down list. Clear if you do not want to implement aggressive merging.
Merges FP Constants	Check to let the linker automatically merge floating-point constants. Clear if you do not want to enable floating-point constants for automatic merging.
Other Flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.7.4.4 PowerPC Linker > Output

Use this panel to specify the output settings for the PowerPC linker.

The following table lists and describes the output settings for PowerPC linker.

Table 3-164. Tool Settings - PowerPC Linker > Output

Option	Description
Output Type	User choose to build an application, a library, or a partial link project.
Optimize Partial Link	Check to specify direct downloading of the partial link output. Clear if you want the output file to remain as if you passed the -r argument in the command line. This checkbox appears only if the **Output Type** drop-down list specifies **Partial Link** .
Deadstrip Unused Symbols	Check to let the linker deadstrip all unused symbols. This reduces program size, by removing symbols that neither the main entry point or force-active entry points reference. Clear if

Table continues on the next page...

Table 3-164. Tool Settings - PowerPC Linker > Output (continued)

Option	Description
	you do not want the linker to deadstrip unused symbols. This checkbox appears only if the **Optimize Partial Link** is checked.
Require Resolved Symbols	Check if the linker must resolve all symbols in the partial link and if your RTOS does not allow unresolved symbols. Clear if the linker does not have to resolve all symbols of the partial link. This checkbox appears only if the **Optimize Partial Link** is checked.
Heap Size (k)	Specify kilobytes of RAM allocated for the heap, which your program uses if it calls malloc or new. Combined heap/stack allocation must not exceed available RAM. This checkbox appears only if the **Output Type** drop-down list specifies **Application** .
Stack Size (k)	Specify kilobytes of RAM allocated for the stack. Combined heap/stack allocation must not exceed available RAM. This checkbox appears only if the **Output Type** drop-down list specifies **Application** .
Interpreter	The linker to use an interpreter file.
Generate Link Map	Check to let the linker generate a link map - showing every object/function definition and address, memory map of sections, and values of linker-generated symbols. Activates subordinate checkboxes. Clear if you do not want the linker to generate a map file. If you used a non-CodeWarrior compiler to build the relocatable file, the map file also lists unused but unstripped symbols. Map files have the extension .MAP
List Closure	Check if you want the map file list all functions that the program starting point calls. Clear if you do not want the map file list functions that the program starting point calls. This checkbox is active only if the **Generate Link Map** checkbox is checked.
List Unused Objects	Check if you want the map file to list unused objects; useful for revealing that objects you expected to be used are not. Clear if you do not want the map to list unused objects. This checkbox is active only if the **Generate Link Map** checkbox is checked.
List DWARF Objects	Check if you want the list map lists all DWARF debugging objects in section area. Clear if you do not want the map file to list DWARF debugging objects. This checkbox is active only if the **Generate Link Map** checkbox is checked.
Generate Binary File	Enbales you to generate one or more raw binary files. By default, no binary file is generated.
Generate S-Record File	Check to generate an S3 S-record file, based on the application object image and activates the subordinate elements. Clear if you do not want to generate an S-record file. The name extension of the S-record file is .mot
Sort S-Record	Check to sort generated S-record files in ascending address order. Clear if you do not want to sort the S-record files. This checkbox is active only if the **Generate S-Record File** checkbox is checked.

Table continues on the next page...

Table 3-164. Tool Settings - PowerPC Linker > Output (continued)

Option	Description
Max S-Record Length	Specifies maximum S-record length (256 bytes or fewer) for the system. (For a non-CodeWarrior tool, you may need to reduce this value.) This checkbox is active only if the **Generate S-Record File** checkbox is checked.
EOL Character	Specifies the end-of-line character for the S-record file:. • Mac - <cr> • DOS - <cr> <lf> • Unix - <lf> This drop-down list is active only if the **Generate S-Record File** checkbox is checked.
Generate Warning Messages	Check if you want the linker to generate warning messages.
Heap Address	Specifies the memory location for program heap where you can enter the RAM address of the bottom of the heap.
Stack Address	Specifies memory location for program stack where you can enter the RAM address for the top of the stack.
Generate ROM Image	Check to enable the **ROM Image Address** and the **RAM Buffer Address of ROM Image** options.
ROM Image Address	Specifies the flash ROM destination address for your binary.
RAM Buffer Address of ROM Image	Specifies the RAM buffer address for the ROM image. This option is active only if a value is specified in the **ROM Image Address** textbox. For the CodeWarrior flash programmer, the ROM image address and the RAM buffer address must be the same.

3.7.5 PowerPC Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the PowerPC Compiler tree control includes the general and the file search path settings.

The following table lists and describes the compiler options for PowerPC.

Table 3-165. Tool Settings - PowerPC Compiler

Option	Description
Command	Shows the location of the compiler executable file. Default value is " `${PAToolsDir}/mwcceppc"`. You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the compiler will be called with.
Expert settings	

Table continues on the next page...

Table 3-165. Tool Settings - PowerPC Compiler (continued)

Option	Description
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`

3.7.5.1 PowerPC Compiler > Preprocessor

Use this panel to specify preprocessor behavior.

The following table lists and describes the preprocessor options for PowerPC.

Table 3-166. Tool Settings - PowerPC Compiler > Preprocessor

Option	Description
Prefix File	Specifies a file automatically included in all project assembly files.
Source encoding	Lets you specify the default encoding of source files. The compiler recognizes Multibyte and Unicode source text. To replicate the obsolete option Multi-Byte Aware, set this option to System or Autodetect. Additionally, options that affect the preprocess request appear in this panel. The options available are as follows: • ASCII • Autodetect • UTF-8 • System • Shift-JIS • EUC-JP • ISO-2022-JP
Defined Macros (-D)	Lists the defined command-line macros.
Undefined Macros (-U)	Lists the undefined command-line macros.

3.7.5.2 PowerPC Compiler > Input

Use this panel to specify additional files the PowerPC Compiler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for PowerPC compiler.

Table 3-167. Tool Settings - PowerPC Compiler > Input Options

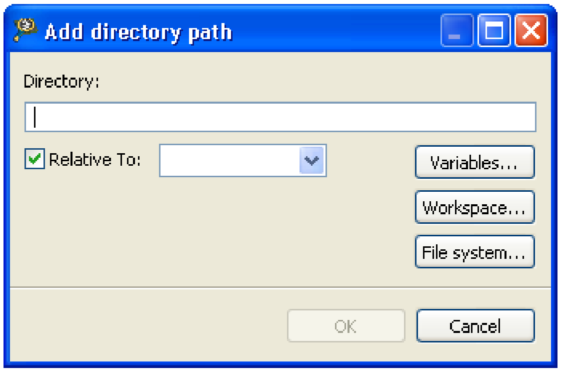
Option	Description
Compile only, Do Not Link	Check if you want to compile only and do not want to link the file.
Do not use MWCIncludes variable	Check if you do not want to use MWCIncludes variable.
Always Search User Paths	Check if you want to always search user paths.
User Path (-i)	Lists the available user paths.
User Recursive Path (-ir)	Appends a recursive access path to the current #include list. This command is global. **Syntax**`-ir pathpath` The recursive access path to append.
System Path (-I- -I)	Lists the available system paths.
System Recursive Path (-I- -ir)	Lists the available system paths recursively.
Disable CW Extensions	Disable the CW features that may be incompatible if user is exporting code libraries form CW to other compiler and/or linkers.

The following table lists and describes the toolbar buttons that help work with the user and system search paths.

Table 3-168. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the search path.
✖	Delete - Click to delete the selected search path.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected search path.
⇧	Move up - Click to move the selected search path one position higher in the list
⇩	Move down - Click to move the selected search path one position lower in the list

The following figure shows the Add directory path dialog box.

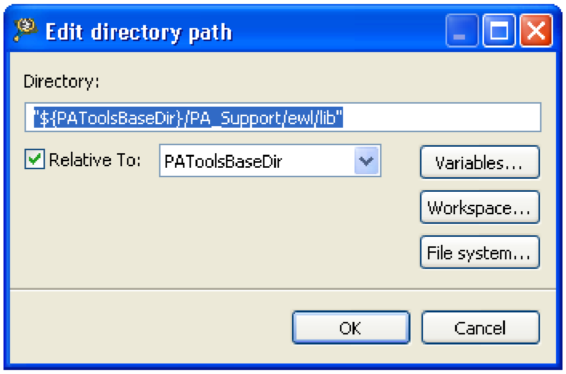


Directory:

☑ Relative To:

Figure 3-31. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



Directory: "${PAToolsBaseDir}/PA_Support/ewl/lib"

☑ Relative To: PAToolsBaseDir

Figure 3-32. Edit directory path Dialog Box

The buttons in the **Add directory** path **and Edit directory path** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **Folder selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse For Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.7.5.3 PowerPC Compiler > Warnings

Use this panel to control how the PowerPC compiler formats the listing file, as well as error and warning messages.

The following table lists and describes the warnings options for PowerPC compiler.

Table 3-169. Tool Settings - PowerPC Compiler > Warnings

Option	Description
Treat All Warnings As Errors	Check to treat all warnings as errors. The compiler will stop if it generates a warning message.
Illegal Pragmas	Check to notify the presence of illegal pragmas.
Possible Errors	Check to suggest possible errors.
Extended Error Checking	Check if you want to do an extended error checking.
Hidden virtual functions	Check to generate a warning message if you declare a non-virtual member function that prevents a virtual function, that was defined in a superclass, from being called and is equivalent to pragma `warn_hidevirtual` and the command-line option `-warnings hidevirtual.`
Implicit Arithmentic Conversions	Check to warn of implict arithmetic conversions.
Implicit Integer to Float Conversions	Check to warn of implict conversion of an integer variable to floating-point type.
Implicit Float to Integer Conversions	Check to warn of implict conversions of a floating-point variable to integer type.
Implicit Signed/Unsigned Conversion	Check to enable warning of implict conversions between signed and unsigned variables.
Pointer/Integral Conversions	Check to enable warnings of conversions between pointer and integers.
Unused Arguments	Check to warn of unused arguments in a function.
Unused Variables	Check to warn of unused variables in the code.
Missing `return' Statement	Check to warn of when a function lacks a return statement.
Expression Has No Side Effect	Check to issue a warning message if a source statement does not change the program's state. This is equivalent to the pragma `warn_no_side_effect`, and the command-line option `-warnings unusedexpr.`
Extra Commas	Check to issue a warning message if a list in an enumeration terminates with a comma. The compiler ignores terminating commas in enumerations when compiling source code that conforms to the ISO/IEC 9899-1999 ("C99") standard and is equivalent to pragma `warn_extracomma` and the command-line `option -warnings extracomma.`
Empty Declarations	Check to warn of empty declarations.
Inconsistent `class' / `struct' Usage	Check to warn of inconsistent usage of class or struct.
Include File Capitalization	Check to issue a warning message if the name of the file specified in a #include "file" directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps` and the command-line option `-warnings filecaps.`

Table continues on the next page...

Table 3-169. Tool Settings - PowerPC Compiler > Warnings (continued)

Option	Description
Check System Includes	Check to issue a warning message if the name of the file specified in a #include <file> directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps_system` and the command-line option `-warnings sysfilecaps`.
Pad Bytes Added	Check to issue a warning message when the compiler adjusts the alignment of components in a data structure and is equivalent to pragma `warn_padding` and the command-line option `-warnings padding`.
Undefined Macro in #if	Check to issues a warning message if an undefined macro appears in #if and #elif directives and is equivalent to pragma `warn_undefmacro` and the command-line option `-warnings undefmacro`.
Non-Inlined Functions	Check to issue a warning message if a call to a function defined with the inline, __inline__, or __inline keywords could not be replaced with the function body and is equivalent to pragma `warn_notinlined` and the command-line option `-warnings notinlined`.

3.7.5.4 PowerPC Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and defines each option of the **Optimization** panel.

Table 3-170. Tool Settings - PowerPC Compiler > Optimization Options

Option	Description
Optimization Level	Specify the optimizations that you want the compiler to apply to the generated object code: • 0 - Disable optimizations. This setting is equivalent to specifying the `-O0` command-line option. The compiler generates unoptimized, linear assembly-language code. • 1 - The compiler performs all target-independent (that is, non-parallelized) optimizations, such as function inlining. This setting is equivalent to specifying the `-O1` command-line option. The compiler omits all target-specific optimizations and generates linear assembly-language code.

Table continues on the next page...

Table 3-170. Tool Settings - PowerPC Compiler > Optimization Options (continued)

Option	Description
	• 2 - The compiler performs all optimizations (both target-independent and target-specific). This setting is equivalent to specifying the `-O2` command-line option. The compiler outputs optimized, non-linear, parallelized assembly-language code. • 3 - The compiler performs all the level 2 optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the `-O3` command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level 2 optimizations. • **4** - The compiler performs all the level 3 optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the `-O4` command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level 3 optimizations.
Speed vs Size	Use to specify an Optimization Level greater than 0. • Speed-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a faster execution speed, as opposed to a smaller executable code size. • Size-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a smaller executable code size, as opposed to a faster execution speed. This setting is equivalent to specifying the `-Os` command-line option.
Inlining	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. Using the `-Oi=c0` option switches off inlining. Functions marked with the `#pragma INLINE` are still inlined. To disable inlining, use the `-Oi=OFF` option. The options available are: • Smart • Auto Inline • Off
Bottom-up Inlining	Check to control the bottom-up function inlining method. When active, the compiler inlines function code starting with the last function in the chain of functions calls, to the first one.

3.7.5.5 PowerPC Compiler > Processor

Use this panel to specify processor behavior. You can specify the file paths and define macros.

The following table lists and defines each option of the Processor panel.

Table 3-171. Tool Settings - PowerPC Compiler > Processor

Option	Description
Struct Alignment	The **Struct Alignment** drop-down list has the default selection PowerPC. To conform with the PowerPC EABI and interoperate with third-party object code, this setting should remain PowerPC. Other settings may lead to reduced performance or alignment violation exceptions. If you choose another setting for Struct Alignment, your code may not work correctly. The options available are as follows: • PowerPC • 68K • 68K 4-byte
Function Alignment	If your board has hardware capable of fetching multiple instructions at a time, you may achieve slightly better performance by aligning functions to the width of the fetch. Use the **Function Alignment** drop-down list to select alignments from 4 bytes (the default) to 128 bytes. These selections corresponds to #pragma function_align.
Relax HW IEEE	The **Relax HW IEEE** checkbox is available only if you select Hardware from the **Floating Point** drop-down list. Check the The **Relax HW IEEE** checkbox to have the compiler generate faster code by ignoring certain strict requirements of the IEEE floating-point standard. These requirements are controlled by the options: • Use Fused Multi-Add/Sub • Generate FSEL Instruction • Assume Ordered Compares
Use Fused Mult-Add/Sub	Check to generate PowerPC Fused Multi-Add/Sub instructions, which result in smaller and faster floating-point code. This may generate unexpected results because of the greater precision of the intermediate values. The generated results are slightly more accurate than those specified by IEEE because of an extra rounding bit between the multiply and the add/subtract.
Generate FSEL Instructions	Check to generate the faster executing FSEL instruction. The FSEL option allows the compiler to optimize the pattern `x = (condition ? y : z)`, where `x` and `y` are floating-point values. FSEL is not accurate for denormalized numbers and may have issues related to unordered compares.
Assume Ordered Compares	Check to allow the compiler to ignore issues with unordered numbers, such as NAN, while comparing floating-point values. In strict IEEE mode, any comparison against NAN, except not-equal-to, returns false. This optimization ignores this provision, thus allowing the following conversion: `if (a <= b)`to `if (b > a)`
Vector Support	Several processors support vector instructions. If you want to allow vector instructions for your processor, select a vector type that your processor supports from the **Vector Support** drop-down list. If you select the **Altivec** option from the **Vector Support** drop-down list, additional options appear. The options available are as follows: • None

Table continues on the next page...

Table 3-171. Tool Settings - PowerPC Compiler > Processor (continued)

Option	Description
	• Altivec • SPE
Generate VRSAVE Instructions	Check the **Generate VRSAVE Instructions** checkbox only when developing for a real-time operating system that supports AltiVec. Checking the **Generate VRSAVE Instructions** checkbox tells the CodeWarrior software to generate instructions to save and restore these vector-register-related values. The VRSAVE register indicates to the operating system which vector registers to save and reload when a context switch happens. The bits of the VRSAVE register that correspond to the number of each affected vector register are set to 1. When a function call happens, the value of the VRSAVE register is saved as a part of the stack frame called the vrsave word. In addition, the function saves the values of any non-volatile vector registers in the stack frame as well, in an area called the vector register save area, before changing the values in any of those registers.
AltiVec Structure Moves	Check if you want the CodeWarrior software to use Altivec instructions when the compiler copies a structure.
Make Strings ReadOnly	Check to store string constants in the read-only .rodata section. Clear to store string constants in the ELF-file data section. The **Make Strings Read Only** checkbox corresponds to #pragma readonly_strings.
Merges String Constants	Check to have the compiler pool strings together from a given file. Clear to let the compiler treat each string as an individual string. The linker can deadstrip unused individual.
Pool Data	Check to instruct the compiler to organize some of the data in the large data sections of .data, .bss, and .rodata so that the program can access it more quickly. This option only affects data that is actually defined in the current source file; it does not affect external declarations or any small data. The linker is normally aggressive in stripping unused data and functions from the C and C++ files in your project. However, the linker cannot strip any large data that has been pooled. If your program uses tentative data, you get a warning that you need to force the tentative data into the common section.
Use Common Section	Check to have the compiler place global uninitialized data in the common section. This section is similar to a FORTRAN Common Block. If the linker finds two or more variables with the same name and at least one of them is in a common section, those variables share the same storage address. If this checkbox is cleared, two variables with the same name generate a link error. The compiler never places small data, pooled data, or variables declared static in the common section.
Use LMW _STMW	Check to have the compiler to use LMW/STMW instructions in the prologue and epilogue of a function when appropriate to store and restore volatile registers.

Table continues on the next page...

Table 3-171. Tool Settings - PowerPC Compiler > Processor (continued)

Option	Description
Inlined Assembler is Volatile	Check to have the compiler treat all asm blocks (including inline asm blocks) as if the volatile keyword was present. This prevents the asm block from being optimized. You can use the .nonvolatile directive to selectively enable optimization on asm blocks, as required.
Instruction Scheduling	Check to optimize the scheduling of instructions for the specific processor you are targeting (determined by which processor is selected in the **Processor** drop-down list.) Enabling the **Instruction Scheduling** checkbox can make source-level debugging more difficult because the source code may not correspond to the execution order of the underlying instructions. It is sometimes helpful to clear this checkbox when debugging, and then check it again once you have finished the bulk of your debugging.
Peephole Optimization	Check to have the compiler perform peephole optimizations. Peephole optimizations are small local optimizations that can reduce several instructions into one target instruction, eliminate some compare instructions, and improve branch sequences. This checkbox corresponds to `#pragma peephole`.
Profiler Information	Check to generate special object code during runtime to collect information for a code profiler. This checkbox corresponds to #pragma profile.
Generate ISEL Instructions (e500/Zen)	Check to have the compiler to emit ISEL instructions. This instruction can improve performance by reducing conditional branching.
Translate PPC Asm to VLE Asm (Zen)	Check to have the compiler to translate the classic PPC inline assembly instructions to the VLE inline assembler for the Zen processor.

3.7.5.6 PowerPC Compiler > C/C++ Language

Use this panel direct the PowerPC compiler to apply specific processing modes to the language source code. You can compile source files with just one collection at a time. To compile source files with multiple collections, you must compile the source code sequentially. After each compile iteration change the collection of settings that the PowerPC compiler uses.

The following table lists and defines each option of the C/C++ Language panel.

Table 3-172. Tool Settings - PowerPC Compiler > C/C++ Language

Option	Description
Force C++ Compilation	Check to translate all C source files as C++ source code. Clear to use the filename's extension to determine whether to use the C or C++ compiler. The entries in the IDE's File Mappings settings panel specify the suffixes that the compiler assigns to each compiler. This checkbox corresponds to the `pragma cplusplus` and the command-line option -lang `c++.`
ISO C++ Template Parser	Check to follow the ISO/IEC 14882-1998 standard for C++ to translate templates, enforcing more careful use of the typename and template keywords. The compiler also follows stricter rules for resolving names during declaration and instantiation. Clear if you do not want the C+++ compiler expect template source code to follow the ISO C++ standard as closely. This checkbox corresponds to the pragma `parse_func_templ` and the command-line option `-iso_templates`
Use Instance Manager	Check to reduce compile time by generating any instance of a C++ template (or non-inlined inline) function only once. Clear to generate a new instance of a template or non-inlined function each time it appears in source code. This checkbox corresponds to control where the instance database is stored using `#pragma instmgr_file` and command-line option `-instmgr`.
Enable C++ Exceptions	Check to generate executable code for C++ exceptions specially when you use the try, throw, and catch statements specified in the ISO/IEC 14882-1998 C++ standard. Clear to generate smaller, faster executable code. The checkbox corresponds to the pragma `exceptions` and the command-line option - `cpp_exceptions.`
Enable RTTI	Check to use of the C++ runtime type information (RTTI) capabilities, including the dynamic_cast and typeid operators. Clear to let the compiler generate smaller, faster object code but do not allow runtime type information operations. The checkbox corresponds to the pragma `RTTI` and the command-line option `-RTTI`.
Enable C++ \`bool' type, \`true' and \`false' Contants	Check to let the C++ compiler recognize the bool type and its true and false values specified in the ISO/IEC 14882-1998 C++ standard. Clear if you do not want the compiler dto recognize this type or its values. The checkbox corresponds to the pragma `bool` and the command-line option `-bool`.
Enable wchar_t Support	Check to let the C++ compiler recognize the `wchar_t` data type specified in the ISO/IEC 14882-1998 C++ standard. Clear if you do not want the compiler to recognize this type or when compiling source code that defines its own wchar_t type. The checkbox corresponds to the pragma `wchar_type` and the command-line option - `wchar_t.`

Table continues on the next page...

Table 3-172. Tool Settings - PowerPC Compiler > C/C++ Language (continued)

Option	Description
EC++ Compatibility Mode	Check if you expect C++ source code files to contain Embedded C++ source code. Clear if the compiler expects regular C++ source code in C++ source files. The checkbox corresponds to the pragma `ecplusplus` and the command-line option `- dialect ec++`.
ANSI Strict	Check if you want the compiler to only recognize source code that conforms to the ISO/IEC 9899-1990 standard for C. Clear if you want the compiler recognize several CodeWarrior extensions to the C language. The checkbox corresponds to the pragma `ANSI_strict` and the command-line option `-ansi strict`.
ANSI Keywords Only	Check to generate an error message for all non-standard keywords. If you must write source code that strictly adheres to the ISO standard, enable this setting. Clear if you want the compiler to recognize only these non-standard keywords: `far`, `inline`, `__inline__`, `__inline`, and `pascal`. The checkbox corresponds to the pragma `only_std_keywords` and the command-line option `-stdkeywords`.
Expand Trigraphs	Check to let the compiler recognize trigraph sequences (ISO/IEC 9899-1990 C, ï¿½5.2.1.1). Clear to ignore trigraph characters. Many common character constants look like trigraph sequences, and this extension lets you use them without including escape characters. The checkbox corresponds to the pragma `trigraphs` and the command-line option `-trigraphs`.
Legacy for-scoping	Check to generate an error message when the compiler encounters a variable scope usage that the ISO/IEC 14882-1998 C++ standard disallows. Clear to let the scope rules specified in ARM. The checkbox corresponds to the pragma `require_prototypes` and the command-line option `-requireprotos`.
Require Prototypes	Check to enforce the requirement of function prototypes. The compiler generates an error message if you define a previously referenced function that does not have a prototype. If you define the function before it is referenced but do not give it a prototype, this setting causes the compiler to issue a warning message. Clear if you do not require prototypes. The checkbox corresponds to the pragma `require_prototypes` and the command-line option `-requireprotos`.
Enable C99 Extensions	Check to let the compiler recognize ISO/IEC 9899-1999 ("C99") language features. Clear if you want the compiler to recognize only ISO/IEC 9899-1990 ("C90") language features. The checkbox corresponds to the pragma `gcc_extensions` and the command-line option `-gcc_extensions`.
Enable GCC Extensions	Check to recognize language features of the GNU Compiler Collection (GCC) C compiler that are supported by CodeWarrior compilers. Clear if you do not want the compiler to recognize GCC extensions. The checkbox corresponds to the pragma `gcc_extensions` and the command-line option `-gcc_extensions`.

Table continues on the next page...

Table 3-172. Tool Settings - PowerPC Compiler > C/C++ Language (continued)

Option	Description
Enum Always Int	Check to use signed integers to represent enumerated constants. Clear to use smallest possible integer type to represent enumerated constants. The checkbox corresponds to the pragma `enumsalwaysint` and the command-line option `-enum`.
Use Unsigned Chars	Check to treat char declarations as unsigned char declarations. Clear to treat char declarations as signed char declarations. The checkbox corresponds to the pragma `unsigned_char` and the command-line option `-char unsigned`.
Pool Strings	Check to collect all string constants into a single data section in the object code it generates. Clear to create a unique section for each string constant. The checkbox corresponds to the pragma `pool_strings` and the command-line option `-strings pool`.
Reuse	Check to store only one copy of identical string literals. Clear to store each string literal separately. The checkbox corresponds to the opposite of the pragma `dont_reuse_strings` and the command-line option `-string reuse`.
IPA	Specifies the Interprocedural Analysis (IPA) policy. • **Off** - No interprocedural analysis, but still performs function-level optimization. Equivalent to the "no deferred inlining" compilation policy of older compilers. • **File** - Completely parse each translation unit before generating any code or data. Equivalent to the "deferred inlining" option of older compilers. Also performs an early dead code and dead data analysis in this mode. Objects with unreferenced internal linkages will be dead-stripped in the compiler rather than in the linker. The checkbox corresponds to the command line option `-ipa`.
Other flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.7.6 PowerPC Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler. Additionally, the Assembler tree control includes the general and include file search path settings.

The following table lists and defines each option of the PowerPC Assembler panel.

Table 3-173. Tool Settings - PowerPC Assembler

Option	Description
Command	Shows the location of the assembler executable file. Default value is `"${PAToolsDir}/mwasmeppc"`. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`
Command line pattern	

3.7.6.1 PowerPC Assembler > Input

Use this panel to specify additional files the PowerPC Assembler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for PowerPC assembler.

Table 3-174. Tool Settings - PowerPC Assembler > Input

Option	Description
Always Search User Paths	CodeWarrior searches only the system path when looking for a file included with #include <...>. To have it searches user path as well, check this box. Note: #include "..." will always search both sets of paths.
User Path (-i)	Lists the available user paths.
User Recursive Path (-ir)	Lists the available user paths recursively.
System Path (-I- -I)	Lists the available system paths.
System Recursive Path (-I- -ir)	Lists the available system paths recursively.

The following table lists and describes the toolbar buttons that help work with the user and system search paths.

Table 3-175. Search Paths Toolbar Buttons

Button	Description
	Add - Click to open the **Add directory path** dialog box and specify the search path.

Table continues on the next page...

Table 3-175. Search Paths Toolbar Buttons (continued)

Button	Description
✖	Delete - Click to delete the selected search path.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected search path.
⇧	Move up - Click to move the selected search path one position higher in the list
⇩	Move down - Click to move the selected search path one position lower in the list

The following figure shows the Add directory path dialog box.

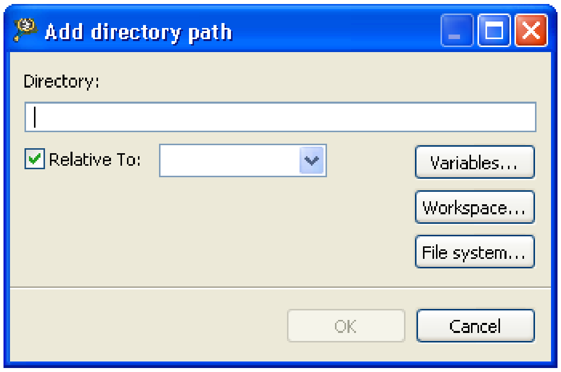
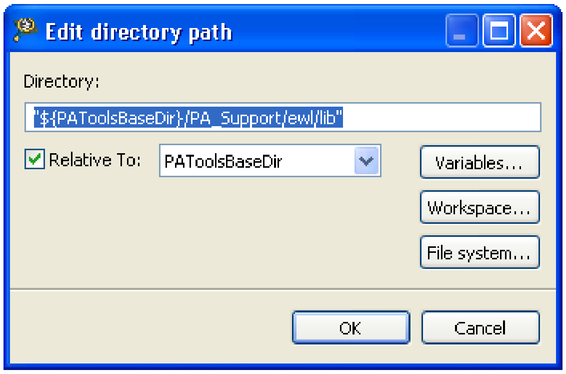


Figure 3-33. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



Directory: "${PAToolsBaseDir}/PA_Support/ewl/lib"

☑ Relative To: PAToolsBaseDir

Figure 3-34. Edit directory path Dialog Box

The buttons in the **Add directory path** and **Edit directory path** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.

- Workspace **-** Click to display the **Folder Selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.7.6.2 PowerPC Assembler > General

Use this panel to specify the general assembler behavior.

The following table lists and describes the general assembler options for PowerPC.

Table 3-176. Tool Settings - PowerPC Assembler > General

Option	Description
Labels Must End With `:'	Check if the source-file labels must end with colon characters. Clear if the source-file labels need not end with colon characters.
Directives Begin With `.'	Check if the assembly directives must begin with period characters. Clear if the assembly directives need not begin with period characters.
Case Sensitive Identifier	Check if casing matters in identifiers. Clear if the assembler ignores case in identifiers.
Allow Space in Operand Field	Check if spaces are allowed in fields. Clear is spaces are not allowed in fields.
GNU Compatible Syntax	Check if your application does use GNU-compatible syntax. This compatibility allows: • Redefining all equates, regardless if from the .equ or .set directives. • Ignoring the .type directive. • Treating undefined symbols as imported. • Using GNU-compatible arithmetic operators - symbols < and > mean left-shift and right-shift instead of less than and greater than; the symbol ! means bitwise-or-not rather than logical not • Using GNU-compatible precedence rules for operators • Implementing GNU-compatible numeric local labels, from 0 to 9 • Treating numeric constants beginning with 0 as octal • Using semicolons as statement separators • Using a single unbalanced quote for character constants - for example, .byte 'a. Clear to indicate that your application does not use GNU-compatible syntax.

Table continues on the next page...

Table 3-176. Tool Settings - PowerPC Assembler > General (continued)

Option	Description
Generate Listing File	Check to let assembler generate a listing file that includes files source, line numbers, relocation information, and macro expansions. Clear if no listing file is specified.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.

3.7.7 PowerPC Disassembler

Use this panel to specify the command, options, and expert settings for PowerPC Disassembler.

The following table lists and describes the PowerPC disassembler options.

Table 3-177. Tool Settings - PowerPC Disassembler

Option	Description
Command	Shows the location of the disassembler executable file. Default value is `powerpc-linux-gnu-as`. You can specify additional command line options for the disassembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert settings	Shows the command line pattern; default is `"${PAGCCINSTALLDIR}/${COMMAND}" ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT}${INPUTS}`
Command line pattern	

3.7.7.1 PowerPC Disassembler > Disassembler Settings

Use this panel to control how the disassembler formats the listing file, as well as error and warning messages. You can specify verbosity of messages, whether to show headers, core modules, extended mnemonics, addresses, object or source code, ldata modules, exception tables, and debug information.

The following table lists and describes the PowerPC disassembler settings.

Table 3-178. Tool Settings - PowerPC Disassembler > Disassembler Settings

Option	Description
Show Headers	Check to display headers in the listing file; disassembler writes listing headers, titles, and subtitles to the listing file
Show Symbol and String Tables	Check to display symbol and string tables directives to the listing file
Show Core Modules	Check to show core modules in the listing file
Show Extended Mnemonics	Check to show the extended mnemonics in the listing file
Show Source Code	Check to show the source code in the listing file
Only Show Operand and Mnemonics	Check this checkbox to have the disassembler list the offset for any functions in the disassembled module.
Show Data Modules	Check to show the data modules in the listing file
Disassemble Exception Tables	Check to disassemble exception tables in the listing file
Show DWARF Info	Check to have the disassembler include DWARF symbol information in the disassembled output. Checking this checkbox makes the **Relocate DWARF Info** checkbox available.
Relocate DWARF Info	Check to relocate object and function addresses in the DWARF information.
Verbose	Check to shows each command line that it passes to the shell, along with all progress, error, warning, and informational messages that the tools emit

3.7.8 PowerPC Preprocessor

Use this panel to specify preprocessor behavior and define macros.

The following table lists and describes the preprocessor options for PowerPC.

Table 3-179. Tool Settings - PowerPC Preprocessor

Option	Description
Command	Shows the location of the preprocessor executable file. Default value is `"${PAToolsDir}/mwcceppc"`. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert settings Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${INPUTS}`

3.7.8.1 PowerPC Preprocessor > Preprocessor Settings

Use this panel to specify preprocessor behavior.

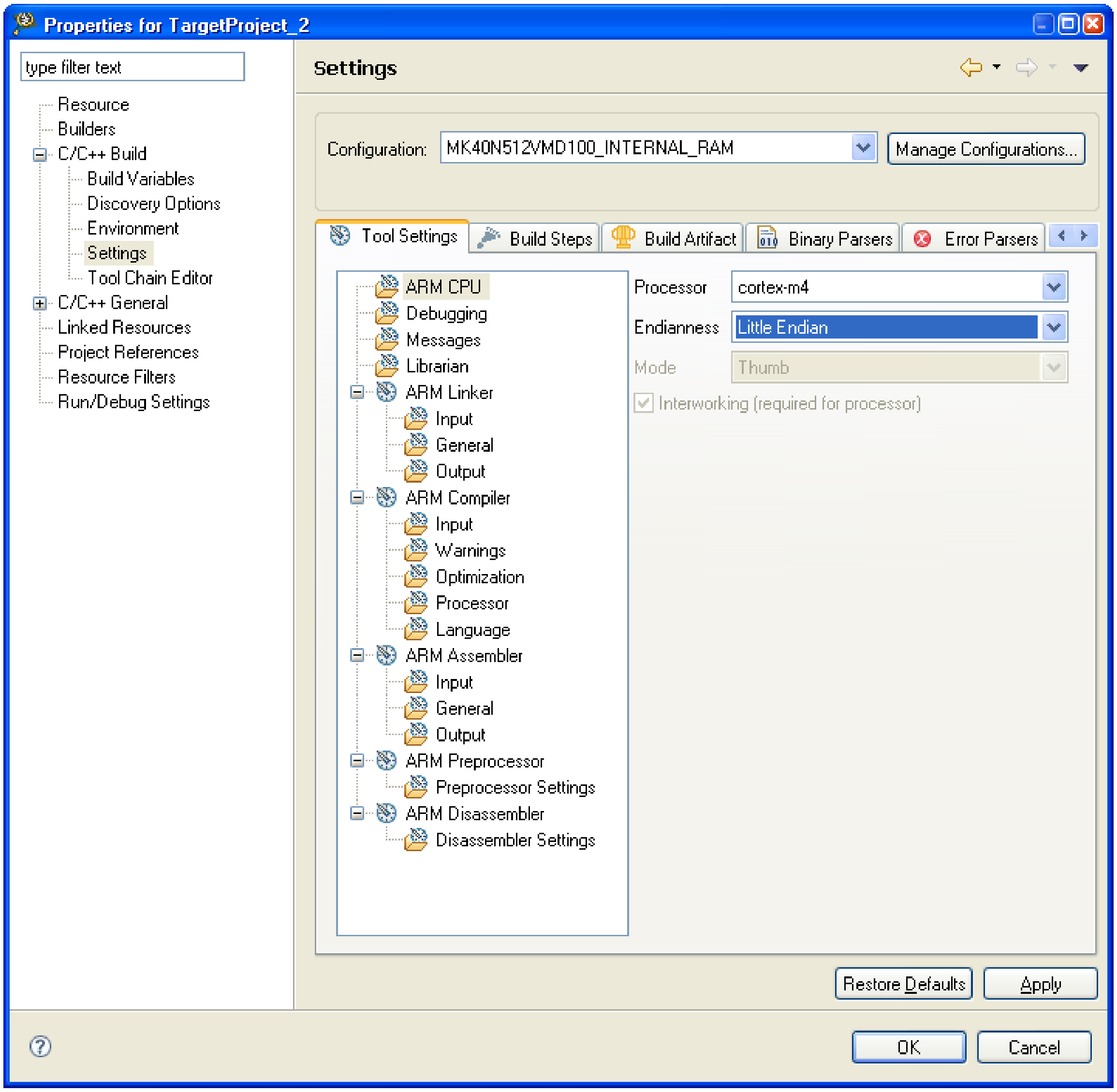
The following table lists and describes the preprocessor settings options for PowerPC.

Table 3-180. Tool Settings - Preprocessor Settings > Preprocessor Options

Option	Description
Mode	Select the mode from the drop-down list.
Emit file change	Check to notify file changes (or #line changes) appear in the output.
Emit #pragmas	Check to show pragma directives in the preprocessor output. Essential for producing reproducible test cases for bug reports.
Show full path	Check to display file changes in comments (as before) or in #line directives.
Keep comment	Check to display comments in the preprocessor output.
Use #line	Check to display file changes in comments (as before) or in #line directives.
Keep whitespace	Check to copy whitespaces in preprocessor output. This is useful for keeping the starting column aligned with the original source, though the compiler attempts to preserve space within the line. This does not apply when macros are expanded.

3.8 Build Properties for ARM (Kinetis)

The **Properties for** <*project*> window shows the corresponding build properties for ARM CPU project (Figure 3-35).



Configuration: MK40N512VMD100_INTERNAL_RAM

Processor: cortex-m4

Endianness: Little Endian

Mode: Thumb

☑ Interworking (required for processor)

Figure 3-35. Build Properties - Kinetis

The following table lists the build properties specific to developing software for Kinetis.

The properties that you specify in the **Tool Settings** panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> window.

Table 3-181. Build Properties for ARM (Kinetis)

Build Tool	Build Properties Panels
ARM CPU	ARM CPU
Debugging	Debugging
Messages	Messages
Librarian	Librarian
ARM Linker	ARM Linker > Input
	ARM Linker > General

Table continues on the next page...

Table 3-181. Build Properties for ARM (Kinetis) (continued)

Build Tool	Build Properties Panels
	ARM Linker > Output
ARM Compiler	ARM Compiler > Input
	ARM Compiler > Warnings
	ARM Compiler > Optimization
	ARM Compiler > Processor
	ARM Compiler > Language
ARM Assembler	ARM Assembler > Input
	ARM Assembler > General
	ARM Assembler > Output
ARM Preprocessor	ARM Preprocessor > Preprocessor Settings
ARM Disassembler	ARM Disassembler > Disassembler Settings

3.8.1 ARM CPU

Use this panel to specify the CPU type, and the encoding that the architecture uses. The build tools (compiler, linker, and assembler) then use the properties set in this panel to generate CPU-specific code.

The following table lists and describes the ARM CPU options.

Table 3-182. Tool Settings - ARM CPU Options

Option	Description
Processor	Lists the processor families supported by the ARM compiler. When you select a processor from this list, the compiler generates code that makes use of any of its hardware features or special instructions. For more detailed information on the features of each processor, refer to its reference manual document.
Floating Point	Specifies handling method for floating point operations: • **Software** - C runtime library code emulates floating-point operations. • **Hardware vfpv4** - Processor hardware performs floating point operations; only appropriate for processors that have floating-point units. Default: Software For software selection, your project must include the appropriate FP_ARM C runtime library file. Grayed out if your target processor lacks an FPU.
Endianness	Lists the byte order for the output file.
Mode	Select the mode from the drop-down list.

Table continues on the next page...

Table 3-182. Tool Settings - ARM CPU Options (continued)

Option	Description
Interworking (required for processor)	Check to if you write ARM code that you want to interwork with Thumb code, or Thumb code that you want to interwork with ARM code. The only functions that need to be compiled for interworking are the functions that are called from the other state. The linker generates suitable interworking veneers when it links the assembler output. Clear if you write ARM code that you do not want to interwork with Thumb code, or Thumb code that you do not want to interwork with ARM code. If you check this checkbox, you must ensure that your code uses the correct interworking return instructions. The IDE enables this setting only for architectures and processors that support ARM/Thumb interworking.

3.8.2 Debugging

Use this panel to specify the options whether to generate symbolic information for debugging the build target.

The following table lists and describes the debugging options.

Table 3-183. Tool Settings - Debugging Options

Option	Description
Generate Debug Information	Check to generate symbolic information for debugging the build target.

3.8.3 Messages

Use this panel to specify the options whether to generate symbolic information for debugging the build target.

The following table lists and describes the message options.

Table 3-184. Tool Settings - Messages Options

Option	Description
Maximum Number of Errors	Specify the number of errors allowed until the application stops processing.
Maximum Number of Warnings	Specify the maximum number of warnings.

3.8.4 Librarian

Use this panel to select options whether the linker will identify standard libraries.

The following table lists and describes the librarian options.

Table 3-185. Tool Settings - Librarian Options

Option	Description
Enable automatic library configurations	Select to let the compiler identify standard libraries.
Model	Select a standard complying or EWL model from the drop-down list. EWL lets you precisely define the I/O operations. EWL drastically reduces the size of executables as you explicitly select the appropriate I/O behavior. Options are: e wl, c9x, ewl_c++, and c9x_c++.
Print formats	Select the print formats from the drop-down list. The available options are: int, int_FP, int_LL, and int_LL_FP.
Scan formats	Select the scan formats from the drop-down list. The available options are: int, int_FP, int_LL, and int_LL_FP.
IO Mode	Select the input-output mode from the drop-down list. The available options are: raw and buffered.

3.8.5 ARM Linker

Use this panel to specify ARM linker behavior. You can specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the input, general, and output settings.

The following table lists and describes the linker options for ARM.

Table 3-186. Tool Settings - ARM Linker Options

Option	Description
Command	Shows the location of the linker executable file. Default value is: "${CF_ToolsDir}/mwldarm" . You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the ARM linker will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}.

3.8.5.1 ARM Linker > Input

Use this panel to specify files the ARM Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for ARM.

Table 3-187. Tool Settings - ARM Linker > Input Options

Option	Description
No Standard Library	Select if there is no standard library attached.
Dead-strip unused code	Determines whether to pool constants from all functions in a file.
Suppress Link Warnings	Prevents the IDE from displaying linker warning messages.
Linker Command File	Consists of three kinds of segments, which must be in this order: A memory segment, which begins with the MEMORY{} directive Optional closure segments, which begin with the FORCE_ACTIVE{}, KEEP_SECTION{}, or REF_INCLUDE{} directives A sections segment, which begins with the SECTIONS{} directive
Entry Point	Specifies the program starting point: the first function the debugger uses upon program start; default: __thumb_startup. This default function is in file ARM__thumb_startup.c. It sets up the ARM EABI environment before code execution. Its final task is calling main().
Library Search Paths	Specifies the search pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click Workspace or File system, then use the subsequent dialog box to browse to the correct location.
Additional Library Files	Specifies the pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click Workspace or File system, then use the subsequent dialog box to browse to the correct location.
Force Active Symbols	Disables deadstripping for particular symbols, enter the symbol names in the Force Active Symbols text box of the ARM Linker Panel.

3.8.5.2 ARM Linker > General

Use this panel to specify the general linker behavior.

The following table lists and describes the general linker options for ARM.

Table 3-188. Tool Settings - ARM Linker > General Options

Option	Description
Other Flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.8.5.3 ARM Linker > Output

Use this panel to specify the output settings for the ARM linker.

The following table lists and describes the output settings for ARM linker.

Table 3-189. Tool Settings - ARM Linker > Output Options

Option	Description
Output Type	Select application as Application (default), Library, or Partial Linking.
Generate Link Map	Check to generate link map.
List Unused Symbols in Map	Check to list unused symbols; appears grayed out if the Generate Link Map checkbox is not checked.
Show Transitive Closure in Map	Check show transitive closure; appears grayed out if the Generate Link Map checkbox is not checked.
Keep Map on Failure	Check to keep the linker generated map in case of failure.
Generate Symbol Table	Check to generate symbol table.
Sort symbols by Address	Check to sort symbols by address; appears grayed out if the Generate Symbol Table checkbox is not checked.
Mapping Symbols First	Check to map symbols first; appears grayed out if the Generate Symbol Table checkbox is not checked.
Generate S-Record File	Check to generate a S-record file.
Max S-Record Length	Specify the maximum length for S-record; appears grayed out if the Generate S-Record File checkbox is not checked. The default value is 252.
S-Record EOL Character	Specify the end-of-line character; appears grayed out if the Generate S-Record File checkbox is not checked. The default value is DOS (\r\n).
Generate X-Record File	Check to generate a X-record file.
Max X-Record Length	Specify the maximum value for X-record; appears grayed out if the Generate X-Record checkbox is not checked.

3.8.6 ARM Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the ARM Compiler tree control includes the general, include file search path settings.

The following table lists and describes the compiler options for ARM.

Table 3-190. Tool Settings - ARM Compiler Options

Option	Description
Command	Shows the location of the compiler executable file. Default value is: `"${ARM_ToolsDir}/mwccarm" -gccinc`. You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the ARM compiler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT}-c ${INPUTS}`.
Command line pattern	

3.8.6.1 ARM Compiler > Input

Use this panel to specify additional files the ARM Compiler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for ARM compiler.

Table 3-191. Tool Settings - ARM Compiler > Input Options

Option	Description
Allow Macro Redefinition	Enables to redefine the macros with the `#define` directive without first undefining them with the `#undef` directive.
Define AEABI Portability	Defines the _AEABI_PORTABILITY_LEVEL to 1. The option ensures that C library dependencies produced by this compiler are ARM EABI compliant. For more information on the ARM EABI and associated compatibility models, the user should consult the ARM, Ltd. website.
Prefix File	Specifies a prefix file that you want the compiler to include at the top of each file.
Source File Encoding	Enables you to specify the default encoding of the source file.

Table continues on the next page...

Table 3-191. Tool Settings - ARM Compiler > Input Options (continued)

Option	Description
Include User Search Paths (-i)	Enables you to add new directories to the list of directories where the user files are searched. Note: Manually edit the absolute paths if the project is moved to a new location. The path will be correct, however, for the location in which the project was originally created. Furthermore, it is a good idea to open the settings of the new project and click 'Apply' to avoid the problem of disappearing user-added include paths.
Include User Recursive Search Paths (-ir)	Enables you to add new directories to the list of directories recursively where the user files are searched.
Include System Search Paths (-I- -I)	Enables you to add new directories to the list of directories where the system files are searched.
Include System Recursive Search Paths (-I- -ir)	Enables you to add new directories to the list of directories recursively where the system files are searched.
Defined Macros	Lists the defined command-line macros.
Undefined Macros	Lists the undefined command-line macros.

3.8.6.2 ARM Compiler > Warnings

Use this panel to control how the ARM compiler formats the listing file, as well as error and warning messages.

The following table lists and describes the warnings options for ARM compiler.

Table 3-192. Tool Settings - ARM Compiler > Warnings Options

Option	Description
Treat All Warnings As Errors	Check to treat all warnings as errors. The compiler will stop if it generates a warning message.
Enable Warnings	Select the level of warnings you want reported from the compiler. Custom lets you to select individual warnings. Other settings select a pre-defined set of warnings.
Illegal #Pragmas (most)	Check to notify the presence of illegal pragmas.
Possible Unwanted Effects (most)	Check to notify most of the possible errors.
Extended Error Checks (most)	Check if you want to do an extended error checking.
Hidden Virtual Functions (most)	Check to generate a warning message if you declare a non-virtual member function that prevents a virtual function, that was defined in a superclass, from being called and is equivalent to pragma `warn_hidevirtual` and the command-line option `-warnings hidevirtual.`
Implicit Arithmentic Conversions (all)	Check to warn of implict arithmetic conversions.
Implicit Signed/Unsigned Conversion (all)	Check to enable warning of implict conversions between signed and unsigned variables.

Table continues on the next page...

Table 3-192. Tool Settings - ARM Compiler > Warnings Options (continued)

Option	Description
Implicit Float to Integer Conversions (all)	Check to warn of implict conversions of a floating-point variable to integer type.
Implicit Integer to Float Conversions (all)	Check to warn of implict conversion of an integer variable to floating-point type.
Pointer/Integer Conversions (most)	Check to enable warnings of conversions between pointer and integers.
Relax Implicit Integer to Interger Arithmetic Conversions	Check to warn when the compiler applies implicit conversion of integer.
Unused Arguments (most)	Check to warn of unused arguments in a function.
Unused Variables (most)	Check to warn of unused variables in the code.
Unused Result From Non-Void-Returning Function (full)	Check to warn of unused result from non-void-returning functions.
Missing `return' Value in Non-Void-Returning Function (most)	Check to warn of when a function lacks a return statement.
Expression has no Side Effect (most)	Check to issue a warning message if a source statement does not change the program's state. This is equivalent to the pragma `warn_no_side_effect`, and the command-line option `-warnings unusedexpr.`
Extra Commas (most)	Check to issue a warning message if a list in an enumeration terminates with a comma. The compiler ignores terminating commas in enumerations when compiling source code that conforms to the ISO/IEC 9899-1999 ("C99") standard and is equivalent to pragma `warn_extracomma` and the command-line `option -warnings extracomma.`
Empty Declarations (most)	Check to warn of empty declarations.
Inconsistent `class' / `struct' Usage (most)	Check to warn of inconsistent usage of class or struct.
Incorrect Capitalization in #include (most)	Check to issue a warning message if the name of the file specified in a #include "file" directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps` and the command-line option `-warnings filecaps.`
Incorrect Capitalization in System #Includes (most)	Check to issue a warning message if the name of the file specified in a #include <file> directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps_system` and the command-line option `-warnings sysfilecaps.`
Pad Bytes Added (full)	Check to issue a warning message when the compiler adjusts the alignment of components in a data structure and is equivalent to pragma `warn_padding` and the command-line option `-warnings padding.`
Undefined Macro in #if/#elif (full)	Check to issues a warning message if an undefined macro appears in #if and #elif directives and is equivalent to pragma `warn_undefmacro` and the command-line option `-warnings undefmacro.`
Non-Inlined Functions (full)	Check to issue a warning message if a call to a function defined with the inline, __inline__, or __inline keywords could not be replaced with the function body and is equivalent to pragma `warn_notinlined` and the command-line option `-warnings notinlined.`

Table continues on the next page...

Table 3-192. Tool Settings - ARM Compiler > Warnings Options (continued)

Option	Description
Token not Formed by ## Operator (most)	Check to enable warnings for the illegal uses of the preprocessor's token concatenation operator (##). It is equivalent to the pragma `warn_illtokenpasting on`.

3.8.6.3 ARM Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and defines each option of the **Optimization** panel.

Table 3-193. Tool Settings - ARM Compiler > Optimization Options

Option	Description
Optimization Level (-opt)	Specify the optimizations that you want the compiler to apply to the generated object code: 0-Disable optimizations. This setting is equivalent to specifying the `-O0` command-line option. The compiler generates unoptimized, linear assembly-language code. 1-The compiler performs all target-independent (that is, non-parallelized) optimizations, such as function inlining. This setting is equivalent to specifying the `-O1` command-line option. The compiler omits all target-specific optimizations and generates linear assembly-language code. 2-The compiler performs all optimizations (both target-independent and target-specific). This setting is equivalent to specifying the `-O2` command-line option. The compiler outputs optimized, non-linear, parallelized assembly-language code. 3-The compiler performs all the level 2 optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the `-O3` command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level 2 optimizations. 4- The compiler performs all the level 3 optimizations. This setting is equivalent to specifying the `-O4 command-line option.` At this level, the compiler adds repeated subexpression elimination and loop-invariant code motion.
Speed Vs Size	Use to specify an Optimization Level greater than 0.

Table continues on the next page...

Table 3-193. Tool Settings - ARM Compiler > Optimization Options (continued)

Option	Description
	• Speed-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a faster execution speed, as opposed to a smaller executable code size. • Size-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a smaller executable code size, as opposed to a faster execution speed. This setting is equivalent to specifying the `-Os` command-line option.
Inter-Procedural Analysis	Control whether the compiler views single or multiple source files at compile time. • Off-Compiler compiles one file at a time. The functions are displayed in order as they appear in the source file. An object file is created for each source. • File-The compiler sees all the functions and data in a translation unit (source file) before code or data is generated. This allows inlining of functions that may not have been possible in -ipa off mode.
Inlining	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. Using the -Oi=c0 option switches off inlining. Functions marked with the #pragma INLINE are still inlined. To disable inlining, use the -Oi=OFF option. • Smart - Inlines function declared with the inline qualifier. This is default. • Auto Inline - Inlines small function even if they are not declared with the inline qualifier. • Off - No functions are inlined.
Bottom-up Inlining	Check to control the bottom-up function inlining method. When active, the compiler inlines function code starting with the last function in the chain of functions calls, to the first one.

3.8.6.4 ARM Compiler > Processor

Use this panel to specify processor behavior. You can specify the file paths and define macros.

The following table lists and defines each option of the Processor panel.

Table 3-194. Tool Settings - ARM Compiler > Processor Options

Option	Description
Allow Semihosting	This option enables users to ARM semihosting features in their applications (such as console output).
Enable ARM Shared Library Architecture Support	This option is disabled for this product.

Table continues on the next page...

Table 3-194. Tool Settings - ARM Compiler > Processor Options (continued)

Option	Description
Pool Constants and Disable Dead-Stripping	This option combines the literal constant pools of multiple functions in a translation unit. The implication of this is that multiple functions reside in a single ELF section, thus disabling dead-stripping by the linker (each function must reside in a unique ELF section to be dead-stripped).
Generate Code for Profiling	Check to enable the processor generate code for use with a profiling tool. Checking this box corresponds to using the command-line option `-profile`. Clearing this checkbox is equivalent to using the command-line option `-noprofile`.
Position-Independent Code	Equivalent to -pic, Using PIC frees you from having to commit to loading your code at a particular address in memory. This means that the code can move to different memory locations and still work correctly. The generated code that is the same regardless of its load address.
Position-Independent Data	Equivalent to -pid, Using PID frees you from having to commit to loading your data at a particular address in memory. This means that the data can move to different memory locations and still work correctly.
Place Read-Only Strings in .rodata Section	Instructs the compiler to place string constants into the .rodata section.
Use Generic Static Symbol Names	Obfuscates the name of static symbols within binaries (such as libraries) as protective measure against unauthorized persons disassembling the binary. Such a disassembly can reveal the names of static symbols and may expose internal structures and other proprietary details.
Set Max Size before Spill to .sdata (bytes)	Enter the maximum number of bytes (n) of an object length for which the processor uses a .sdata section.

3.8.6.5 ARM Compiler > Language

Use this panel direct the ARM compiler to apply specific processing modes to the language source code. You can compile source files with just one collection at a time. To compile source files with multiple collections, you must compile the source code sequentially. After each compile iteration change the collection of settings that the ARM compiler uses.

The following table lists and defines each option of the Language Settings panel.

Table 3-195. Tool Settings - ARM Compiler > Language Settings Options

Option	Description
Force C++ Compilation	Check to translate all C source files as C++ source code.

Table continues on the next page...

Table 3-195. Tool Settings - ARM Compiler > Language Settings Options (continued)

Option	Description
ANSI Strict	Check to enable C compiler operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ ISO specification to all input files. This setting is equivalent to specifying the `-ansi` command-line option. The compiler issues a warning for each ANSI/ISO extension it finds.
ANSI Keywords Only	**Check to** generate an error message for all non-standard keywords (ISO/IEC 9899-1990 C, ï¿½6.4.1). If you must write source code that strictly adheres to the ISO standard, enable this setting; is equivalent to pragma `only_std_keywords` and the command-line option `-stdkeywords`.
Enable C99 Extensions	**Check to r** ecognize ISO/IEC 9899-1999 ("C99") language features; is equivalent to pragma `c99` and the command-line option - `dialect c99`.
Enable GCC Extensions	**Check to** recognize language features of the GNU Compiler Collection (GCC) C compiler that are supported by CodeWarrior compilers; is equivalent to pragma `gcc_extensions` and the command-line option `-gcc_extensions.`
Enums Always Int	**Check to u** se signed integers to represent enumerated constants and is equivalent to pragma `enumsalwaysint` and the command-line option `-enum`.
Use Unsigned Chars	**Check to t** reat char declarations as unsigned char declarations and is equivalent to pragma `unsigned_char` and the command-line option `-char unsigned.`
Require Function Prototypes	Check to enforce the requirement of function prototypes. The compiler generates an error message if you define a previously referenced function that does not have a prototype. If you define the function before it is referenced but do not give it a prototype, this setting causes the compiler to issue a warning message.
Expand Trigraphs	**Check to** recognize trigraph sequences (ISO/IEC 9899-1990 C, ï¿½5.2.1.1); is equivalent to **pragma** `trigraphs` **and the command-line option** `-trigraphs.`
Enable Exceptions	**Check to** generate executable code for C++ exceptions; is equivalent to pragma `exceptions` and the command-line option - `cpp_exceptions`.
Enable RTTI Support	Check to allow the use of the C++ runtime type information (RTTI) capabilities, including the `dynamic_cast` and `typeid` operators; is equivalent to pragma RTTI and the command-line option -RTTI.
Enable bool support	Check to enable the C++ compiler to recognize the bool type and its true and false values specified in the ISO/IEC 14882-1998 C++ standard.
Enable wchar_t support	**Check to enable** C++ compiler recognize the wchar_t data type specified in the ISO/IEC 14882-1998 C++ standard; is equivalent to pragma `wchar_type` and the command-line option `-wchar_t`.

Table continues on the next page...

Table 3-195. Tool Settings - ARM Compiler > Language Settings Options (continued)

Option	Description
ISO Template Parser	Check to follow the ISO/IEC 14882-1998 standard for C++ to translate templates, enforcing more careful use of the typename and template keywords. The compiler also follows stricter rules for resolving names during declaration and instantiation.
Use Instance Manager	**Check to r** educe compile time by generating any instance of a C++ template (or non-inlined inline) function only once.
Legacy for-scoping	**Check to g** enerate an error message when the compiler encounters a variable scope usage that the ISO/IEC 14882-1998 C++ standard disallows, but is allowed in the C++ language specified in The Annotated C++ Reference Manual ("ARM"); is equivalent to pragma `ARM_scoping` and the command-line option `-for_scoping`.
Reuse Strings	**Check to** store only one copy of identical string literals and is equivalent to opposite of the pragma `dont_reuse_strings` and the command-line `option -string reuse.`
Pool Strings	**Check to** collect all string constants into a single data section in the object code it generates and is equivalent to pragma `pool_strings` and the command-line `option -strings pool.`
Other flags	Specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI. **Note** : To enable CodeWarrior MCU V10.x to generate .lst file for each source file in ARM, you need to specify -S in the **Other Flags** option.

3.8.7 ARM Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler. Additionally, the Assembler tree control includes the general and include file search path settings.

The following table lists and defines each option of the ARM Assembler panel.

Table 3-196. Tool Settings - ARM Assembler Options

Option	Description
Command	Shows the location of the assembler executable file. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	

Table continues on the next page...

Table 3-196. Tool Settings - ARM Assembler Options (continued)

Option	Description
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.

3.8.7.1 ARM Assembler > Input

Use this panel to specify additional files the ARM Assembler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the input options for ARM assembler.

Table 3-197. Tool Settings - ARM Compiler > Input Options

Option	Description
Prefix File	Specify a prefix file that you want the compiler to include at the top of each file.
Enable Debug	Check to automatically generate the debug information for the project.
Always Search Both User and System Paths (-nosyspath)	Performs a search of both the user and system paths, treating #include statements of the form #include <xyz> the same as the form #include "`xyz`".
Include User Search Paths (-i)	Enables you to add new directories to the list of directories where the user files are searched.
Include User Recursive Search Paths (-ir)	Enables you to add new directories to the list of directories recursively where the user files are searched.
Include System Search Paths (-I- -I)	Enables you to add new directories to the list of directories where the system files are searched.
Include System Recursive Search Paths (-I- -ir)	Enables you to add new directories to the list of directories recursively where the system files are searched.

The following table lists and describes the toolbar buttons that help work with the user and system search paths.

Table 3-198. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the search path.
✖	Delete - Click to delete the selected search path.

Table continues on the next page...

Table 3-198. Search Paths Toolbar Buttons (continued)

Button	Description
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected search path.
⇧	Move up - Click to move the selected search path one position higher in the list
⇩	Move down - Click to move the selected search path one position lower in the list

Teh following figure shows the Add directory path dialog box.

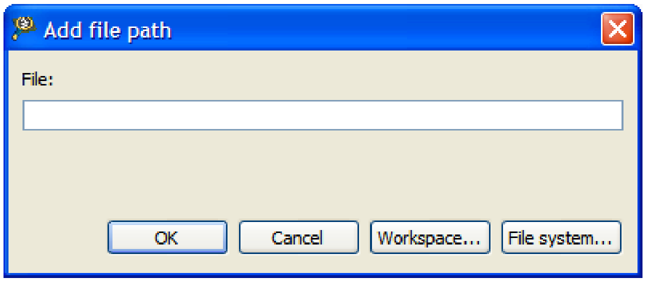
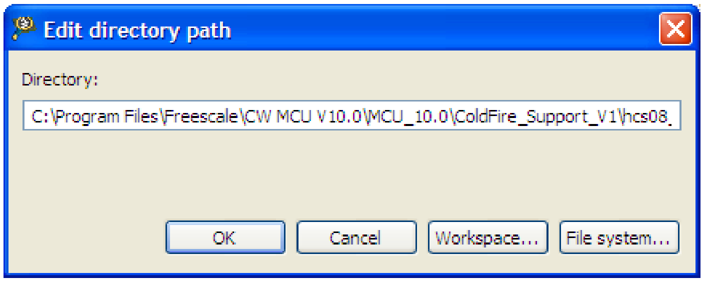


Figure 3-36. Add directory path Dialog Box

The following table shows the Edit directory path dialog box.



Directory:
C:\Program Files\Freescale\CW MCU V10.0\MCU_10.0\ColdFire_Support_V1\hcs08_

Figure 3-37. Edit directory path Dialog Box

The buttons in the **Add directory path and Edit directory path** dialog boxes help work with the object file search paths.

- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.
- Workspace **-** Click to display the **Folder selection** dialog box and specify the object file search path. The resulting path, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **Variables-** Click to display the **Select build variable** dialog box and select the object file variable for search path. The resulting variable in the appropriate list.

3.8.7.2 ARM Assembler > General

Use this panel to specify the general assembler behavior.

The following table lists and describes the general assembler options for ARM.

Table 3-199. Tool Settings - ARM Assembler > General Options

Option	Description
Label Must End With `:'	Clear if system does not require labels to end with colons. By default, the option is checked.
Directives Begin With `.'	Clear if the system does not require directives to start with periods. By default, the option is checked.
Allow Space In Operand Field	Clear to restrict the assembler from adding spaces in operand fields. By default, the option is checked.
Case-Sensitive Identifier	Clear to instruct the assembler to ignore case in identifiers. By default, the option is checked.
Enable GNU Assembler Compatible Syntax	Instructs the assembler to accept GNU-style assembly syntax.
Enable ARM ADS Compatible Syntax	Instructs the assembler to accept ARM ADS compatible syntax extensions.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.

3.8.7.3 ARM Assembler > Output

Use this panel to specify the output settings for the ARM assembler.

The following table lists and describes the output assembler options for ARM.

Table 3-200. Tool Settings - ARM Assembler > Output Options

Option	Description
Generate Listing File	Instructs the assembler to generate a disassembly output file. The disassembly output file contains the file source, along with line numbers, relocation information, and macro expansion.

3.8.8 ARM Preprocessor

Use this panel to specify preprocessor behavior and define macros.

The following table lists and describes the preprocessor options for ARM.

Table 3-201. Tool Settings - ARM Preprocessor Options

Option	Description
Command	Shows the location of the preprocessor executable file. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the preprocessor will be called with.
Expert Settings	Shows the command line pattern; default `${COMMAND} -E${FLAGS} ${INPUTS}`.
Command line pattern	

3.8.8.1 ARM Preprocessor > Preprocessor Settings

Use this panel to specify preprocessor behavior.

The following table lists and describes the preprocessor options for ARM.

Table 3-202. Tool Settings - ARM Preprocessor Options

Option	Description
Emit File/Line Breaks	Check to notify file breaks (or #line breaks) appear in the output.
Keep #pragmas	Check to show pragma directives in the preprocessor output. Essential for producing reproducible test cases for bug reports.
Show Full Path	Check to display file changes in comments (as before) or in #line directives.
Keep Comments	Check to display comments in the preprocessor output.
Emit #line Directives	Check to display file changes in comments (as before) or in #line directives.
Keep Whitespace	Check to copy whitespaces in preprocessor output. This is useful for keeping the starting column aligned with the original source, though the compiler attempts to preserve space within the line. This does not apply when macros are expanded.

3.8.9 ARM Disassembler

Use this panel to specify the command, options, and expert settings for ARM Disassembler.

The following table lists and describes the ARM disassembler options.

Table 3-203. Tool Settings - ARM Disassembler Options

Option	Description
Command	Shows the location of the disassembler executable file. You can specify additional command line options for the disassembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} -dis ${FLAGS} ${INPUTS}`
Command line pattern	

3.8.9.1 ARM Disassembler > Disassembler Settings

Use this panel to control how the disassembler formats the listing file, as well as error and warning messages. You can specify verbosity of messages, whether to show headers, core modules, extended mnemonics, addresses, object or source code, ldata modules, exception tables, and debug information.

The following table lists and describes the ARM disassembler settings.

Table 3-204. Tool Settings - ARM Disassembler Options

Option	Description
Show Headers	Check to display headers in the listing file; disassembler writes listing headers, titles, and subtitles to the listing file.
Show Symbol and String Tables	Check to display symbol and string tables directives to the listing file.
Verbose Info	Check to shows each command line that it passes to the shell, along with all progress, error, warning, and informational messages that the tools emit.
Show Relocations	Check to have the disassembler show information about relocated symbols. Clear to prevent the disassembler from showing information about relocated symbols.
Show Code Modules	Check to show code modules in the listing file.
Show Extended Mnemonics	Check to show the extended mnemonics in the listing file.

Table continues on the next page...

Table 3-204. Tool Settings - ARM Disassembler Options (continued)

Option	Description
Show Addresses and Opcodes	Check to show the addresses and object code in the listing file.
Show Source Code	Check to show the source code in the listing file.
Show Comments	Check to show the comments in the listing file.
Show Data Modules	Check to show the data modules in the listing file.
Disassemble Exception Tables	Check to disassemble exception tables in the listing file.
Show Debug Info	Check to generate symbolic information for debugging the build target.

3.9 Build Properties for ARM Ltd Windows GCC

The **Properties for** <*project*> window shows the corresponding build properties for ARM project that supports the GCC toolchain.

The properties that you specify in the **Tool Settings** panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> window.

The following table lists and describes the settings.

Table 3-205. Build Properties for ARM (GCC)

Build Tool	Build Properties Panels
Target Processor	Target Processor
Debugging	Debugging
Additional Tools	Additional Tools
Librarian	Librarian
ARM Ltd. Windows GCC Assembler	ARM Ltd. Windows GCC Assembler > Preprocessor
	ARM Ltd. Windows GCC Assembler > Directories
	ARM Ltd. Windows GCC Assembler > Warnings
	ARM Ltd. Windows GCC Assembler > Miscellaneous
ARM Ltd. Windows GCC Compiler	ARM Ltd. Windows GCC Compiler > Preprocessor
	ARM Ltd. Windows GCC Compiler > Directories
	ARM Ltd. Windows GCC Compiler > Optimization
	ARM Ltd. Windows GCC Compiler > Warnings
	ARM Ltd. Windows GCC Compiler > Miscellaneous
ARM Ltd. Windows GCC Linker	ARM Ltd. Windows GCC Linker > General
	ARM Ltd. Windows GCC Linker > Libraries
	ARM Ltd. Windows GCC Linker > Miscellaneous

Table continues on the next page...

Table 3-205. Build Properties for ARM (GCC) (continued)

Build Tool	Build Properties Panels
ARM Ltd. Windows GCC Disassembler	ARM Ltd. Windows GCC Disassembler > Disassembler Settings
ARM Ltd. Windows GCC C Preprocessor	ARM Ltd. Windows GCC C Preprocessor > Preprocessor Settings

3.9.1 Target Processor

Use the **Target Processor** panel to specify the processor family for the project. The properties specified on this page are also used by the build tools (compiler, linker, and assembler).

The table below lists and describes the various options available on the **Target Processor** panel.

Table 3-206. Tool Settings - Target Processor

Option	Description
Processor	Use to specify the target processor architecture. The debugger configures the appropriate register views according to the target processor that you specify.
Thumb (-mthumb)	Check to have the processor generate Thumb code instructions.
Thumb interwork (-mthumb-interwork)	Check this option if you are writing ARM code that you want to interwork with Thumb code, or Thumb code that you want to interwork with ARM code. The only functions that need to be compiled for interworking are the functions that are called from the other state. The linker generates suitable interworking veneers when the compiler output is linked.
Endiannes	Use to specify the byte order of the target hardware architecture: • Big - Select if the target processor uses big-endian (BE) byte order (leftmost bytes are most significant: B3 B2 B1 B0). • Little - Select if the target processor uses little-endian (LE) byte order (rightmost bytes are most significant: B0 B1 B2 B3).
Float ABI	Specify the floating-point options. The options avaialable are: Toolchain Default, Library (-mfloat -abi=soft), Library with FP (-mfloat-ai=softfp), and FP instructions (-mfloat-abi=hard).
FPU Type	From the list box, select the architecture or processor that corresponds to your target hardware. • Software-software-based FPU library. • FPA-Floating Point Accelerator format and instructions

Table 3-206. Tool Settings - Target Processor

Option	Description
	- VFP-hardware vector FPU format and instructions compatible with the VFPv1 architecture - VFPV2-hardware vector FPU format and instructions compatible with the VFPv2 architecture The compiler might display error messages or warnings if the selected FPU architecture is not compatible with the target architecture.

3.9.2 Debugging

Use the **Debugging** panel to specify the debugging settings for the project.

The table below lists and describes the various options available on the **Debugging** panel.

Table 3-207. Tool Settings - Debugging

Option	Description
Debug level	Specify the debug levels: - None - No Debug level. - Minimal (`-g1`) - The compiler provides minimal debugging support. - Default (`-g`) - The compiler generates DWARF 1.x-conforming debugging information. - Maximum (`-g3`) - The compiler provides maximum debugging support.
Debug format	Specify the debug formats for the compiler. - dwarf-2 - Generates DWARF 2.x-conforming debugging information. - stabs - Generates STABS-conforming debugging information.
Other debugging flags	Specify the other debugging flags that need to be passed with the compiler
Generate prof information (-p)	Generates extra code to write profile information suitable for the analysis program prof. You must use this option when compiling the source files you want data about, and you must also use it when linking.
Generate gprof information (-pg)	Generates extra code to write profile information suitable for the analysis program gprof. You must use this option when compiling the source files you want data about, and you must also use it when linking.

3.9.3 Additional Tools

Use the **Additional Tools** panel to specify additional settings for the project.

The table below lists and describes the various options available on the **Additional Tools** panel.

Table 3-208. Tool Settings - Additional Tools

Option	Description
Create Flash Image	Check if you want to create a flash image.
Create Extended Listing	Check if you want to enable the support for extended listing.
Print Size	Check to enable the print size.

3.9.4 Librarian

Use the **Librarian** panel to specify library settings for the project.

The table below lists and describes the various options available on the **Librarian** panel.

Table 3-209. Tool Settings - Librarian

Option	Description
Enable automatic library configurations	Select to let the compiler identify standard libraries.
Model	Select a standard complying or EWL model from the drop-down list. EWL lets you precisely define the I/O operations. EWL drastically reduces the size of executables as you explicitly select the appropriate I/O behavior. Options are: `ewl, ewl_c++, c9x, c9x_c++, ewl_hosted, ewl_c++_hosted, c9x_hosted, c9x_c++_hosted, ewl_noio, ewl_c++_noio, c9x_noio, and c9x_c++_noio.`
Print formats	Select the print formats from the drop-down list. The available options are: `int, int_FP, int_LL,` and `int_LL_FP`.
Scan formats	Select the scan formats from the drop-down list. The available options are: `int, int_FP, int_LL,` and `int_LL_FP`.

3.9.5 ARM Ltd. Windows GCC Assembler

Use the **ARM Ltd. Windows GCC Assembler** panel to specify the assembler settings for the project.

The following table lists and describes the various options available on the **ARM Ltd. Windows GCC Assembler** panel.

Table 3-210. Tool Settings - ARM Ltd. Windows GCC Assembler

Option	Description
Command	Shows the location of the assembler executable file. Default: `arm-none-eabi-gcc`
All options	Shows the actual command line the assembler will be called with. Default: `-x assembler-with-cpp -Wall -Wa,-adhlns="$@.lst" -c -fmessage-length=0 -mcpu=cortex-m0 -mthumb -g3 -gstabs`
Expert settings	Shows the expert settings command line parameters. Default: `"${ARMSourceryDir}/${COMMAND}" ${INPUTS} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT}`
Command line pattern	

3.9.5.1 ARM Ltd. Windows GCC Assembler > Preprocessor

Use the **Preprocessor** panel to specify the preprocessor behavior. You can specify whether to search system directories or preprocess only based on the options available in this panel.

The following table lists and describes the various options available on the **Preprocessor** panel.

Table 3-211. ARM Ltd. Windows GCC Assembler > Preprocessor

Option	Description
Use preprocessor (-x assembler)	Check this option to use the preprocessor for the assembler.
Do not search system directories (-nostdinc)	Check this option if you do not want the assembler to search the system directories. By default, this checkbox is clear. The assembler performs a full search that includes the system directories.
Preprocess only (-E)	Check this option if you want the assembler to preprocess source files and not to run the compiler. By default, this checkbox is clear and the source files are not preprocessed.
Defined symbols (-D)	Use this option to specify the substitution strings that the assembler applies to all the assembly-language modules in the build target. Enter just the string portion of a substitution string. The IDE prepends the `-D` token to each string that you enter. For example, entering `opt1 x` produces this result on

Table continues on the next page...

Table 3-211. ARM Ltd. Windows GCC Assembler > Preprocessor (continued)

Option	Description
	the command line: `-Dopt1 x`**Note:** This option is similar to the `DEFINE` directive, but applies to all assembly-language modules in a build target.
Undefined symbols (-U)	Undefines the substitution strings you specify in this panel.

3.9.5.2 ARM Ltd. Windows GCC Assembler > Directories

Use the **Directories** panel to specify the directories paths.

The following table lists and describes the various options available on the **Directories** panel.

Table 3-212. ARM Ltd. Windows GCC Assembler > Directories

Option	Description
Include paths (`-I`)	This option changes the build target's search order of access paths to start with the system paths list. The compiler can search `#include` files in several different ways. You can also set the search order as follows: For include statements of the form `#include"xyz"`, the compiler first searches user paths, then the system paths For include statements of the form `#include<xyz>`, the compiler searches only system paths This option is global.

3.9.5.3 ARM Ltd. Windows GCC Assembler > Warnings

Use the **Warnings** panel to control how the compiler reports the error and warning messages.

The following table lists and describes the various options available on the **Warnings** panel.

Table 3-213. ARM Ltd. Windows GCC Assembler > Warnings

Option	Description
Check syntax only (-fsyntax-only)	Check this option if you want to check the syntax of commands and throw a syntax error.

Table continues on the next page...

Table 3-213. ARM Ltd. Windows GCC Assembler > Warnings (continued)

Option	Description
Pedantic (-pedantic)	Check this option if you want to issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any `-std' option used.
Pedantic warnings as erros (-pedantic-errors)	Check this option if you want to issue all the mandatory diagnostics, and make all mandatory diagnostics into errors. This includes mandatory diagnostics that GCC issues without `-pedantic` but treats as warnings.
Inhibit all warnings (-w)	Check this option if you want to inhibit the display of warning messages.
All warnings (-Wall)	Check this option to turn on all optional warnings which are desirable for normal code. At present this is `-Wcomment, -Wtrigraphs, -Wmultichar` and a warning about integer promotion causing a change of sign in #if expressions. **NOTE** : Many of the preprocessor's warnings are on by default and have no options to control them.
Extra warnings (-Wextra)	Check this option to enable any extra warnings.
Warning as errors (-Werror)	Check this option if you want to make all warnings into hard errors. Source code which triggers warnings will be rejected.

3.9.5.4 ARM Ltd. Windows GCC Assembler > Miscellaneous

Use the **Miscellaneous** panel to specify compiler options.

The following table lists and describes the various options available on the **Miscellaneous** panel.

Table 3-214. ARM Ltd. Windows GCC Assembler > Miscellaneous

Option	Description
Assembler flags	Specify the flags that need to be passed with the assembler.
Enable Assembler Listing	Enables the assembler to create a listing file as it compiles assembly language into object code.
Assembler Listing	Displays the listing file. Default: `-adhlns="$@.lst"`
Support ANSI programs (-ansi)	Check this option if you want the assembler to operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ISO specification to all input files. This setting is equivalent to specifying the - ansi command-line option. The compiler issues a warning for each ANSI/ISO extension it finds. By default this checkbox is clear. The assembler does not operate in strict ANSI mode.

Table continues on the next page...

Table 3-214. ARM Ltd. Windows GCC Assembler > Miscellaneous (continued)

Option	Description
Verbose (-v)	Check this option if you want the IDE to show each command-line that it passes to the shell, along with all progress, error, warning, and informational messages that the tools emit. This setting is equivalent to specifying the -v command-line option. By default this checkbox is clear. The IDE displays just error messages that the compiler emits. The IDE suppresses warning and informational messages.
Other flags	Specifies the assembler flags. Default: `-c -fmessage-length=0`

3.9.6 ARM Ltd. Windows GCC Compiler

Use the **ARM Ltd. Windows GCC Compiler** panel to specify the compiler options that are specific to the ARM (GCC).

NOTE

> The list of tools presented on the **Tool Settings** page can differ, based upon the toolchain used by the project.

The following table lists and describes the various options available on the **ARM Ltd. Windows GCC Compiler** panel.

Table 3-215. Tool Settings - ARM Ltd. Windows GCC Compiler

Option	Description
Command	Shows the location of the assembler executable file. Default: `arm-none-eabi-gcc`
All options	Shows the actual command line the assembler will be called with. Default: `-nostdinc -I"D:\Profiles\b14174\16may2012\Project_1/Project_Headers" -I"D:\Freescale\CW MCU v10.x\eclipse\../Cross_Tools/arm-none-eabi-gcc-4_6_2/bin/../ewl/EWL_C/include" -I"D:\Freescale\CW MCU v10.x\eclipse\../Cross_Tools/arm-none-eabi-gcc-4_6_2/bin/../ewl/EWL_Runtime/include" -O0 -Wall -Wa,-adhlns="$@.lst" -c -fmessage-length=0 -mcpu=cortex-m0 -mthumb -g3 -gstabs`
Expert settings	Shows the expert settings command line parameters. Default: `"${ARMSourceryDir}/${COMMAND}" ${INPUTS} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT}"`
Command line pattern	

3.9.6.1 ARM Ltd. Windows GCC Compiler > Preprocessor

Use the **Preprocessor** panel to specify preprocessor behavior. You can specify whether to search system directories or preprocess only based on the options available in this panel.

The following table lists and describes the various options available on the **Preprocessor** panel.

Table 3-216. ARM Ltd. Windows GCC Compiler > Preprocessor

Option	Description
Do not search system directories (-nostdinc)	Check this option to specify the `-nostdinc` command to the compiler. The compiler does not search the system directories. By *default* this checkbox is clear. The compiler performs a full search that includes the system directories
Preprocess only (-E)	Check this option to specify the `-E` command to the compiler. The compiler tells the command-line tool to preprocess source files. By *default* this checkbox is clear. The compiler does not preprocess source files.
Defined symbols (`-D`)	Use this option to specify the substitution strings that the assembler applies to all the assembly-language modules in the build target. Enter just the string portion of a substitution string. The IDE prepends the `-D` token to each string that you enter. For example, entering `opt1 x` produces this result on the command line: `-Dopt1 x`**Note:** This option is similar to the `DEFINE` directive, but applies to all assembly-language modules in a build target.
Undefined symbols (`-U`)	Undefines the substitution strings you specify in this panel.

3.9.6.2 ARM Ltd. Windows GCC Compiler > Directories

Use the **Directories** panel to specify the directories paths.

The following table lists and describes the various options available on the **Directories** panel.

Table 3-217. ARM Ltd. Windows GCC Compiler > Directories

Option	Description
Include paths (`-I`)	This option changes the build target's search order of access paths to start with the system paths list. The compiler can search `#include` files in several different ways. You can also set the search order as follows: For include statements of the form `#include"xyz"`, the compiler first searches user

Table 3-217. ARM Ltd. Windows GCC Compiler > Directories

Option	Description
	paths, then the system paths For include statements of the form `#include<xyz>`, the compiler searches only system paths This option is global.

3.9.6.3 ARM Ltd. Windows GCC Compiler > Optimization

Use the **Optimization** panel to control compiler optimizations. Compiler optimization can be applied in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the various options available on the **Optimization** panel.

Table 3-218. ARM Ltd. Windows GCC Compiler > Optimization

Option	Description
Optimization Level	Use this option to specify the optimizations that you want the compiler to apply to the generated object code. The options available are: • **None (-O0)** - Disable optimizations. This setting is equivalent to specifying the -O0 command-line option. The compiler generates unoptimized, linear assembly-language code. • **Optimize (-O1)** - The compiler performs all target-independent (that is, non-parallelized) optimizations, such as function inlining. This setting is equivalent to specifying the -O1 command-line option. The compiler omits all target-specific optimizations and generates linear assembly-language code. • **Optimize more (-O2)** - The compiler performs all optimizations (both target-independent and target-specific). This setting is equivalent to specifying the -O2 command-line option. The compiler outputs optimized, non-linear, parallelized assembly-language code. • **Optimize most (-O3)** - The compiler performs all the level 2 optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the -O3 command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level 2 optimizations. • **Optimize for size (-Os)** - Optimize for size. `-Os` enables all `-O2` optimizations that do not typically increase code size. It also performs further

Table continues on the next page...

Table 3-218. ARM Ltd. Windows GCC Compiler > Optimization (continued)

Option	Description
	optimizations designed to reduce code size. This setting is equivalent to specifying the - `Os` command-line option.
Pack structures (-fpack-struct)	Packed data structures are supported in the compiler with the keyword `__packed` or `__attribute__((packed))`. There is no code generation support for accessing un-aligned, packed data members.Users should exercise caution when accessing packed data structures because data might not be aligned.
Short enumerations (-fshort-enums)	Check to use short enumerated constants and is equivalent to `-fshort-enums`.
Function sections (-ffunction-sections)	Check to use function sections and is equivalent to `-ffunction-sections`.
Data sections (-fdata-sections)	Check to use short data sections and is equivalent to `-ffunction-sections`.
Other optimization flags	Specifies individual optimization flag that can be turned ON/OFF based on the user requirements.

3.9.6.4 ARM Ltd. Windows GCC Compiler > Warnings

Use the Warnings panel to control how the compiler reports the error and warning messages. The following table lists and describes the various options available on the Warnings panel.

Table 3-219. ARM Ltd. Windows GCC Compiler > Warnings

Option	Description
Check syntax only (-fsyntax-only)	Check this option if if you want to check the syntax of commands and throw a syntax error.
No common (-fno-common)	Check this option if if you want to issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any `-std' option used.
Pedantic (-pedantic)	Check if you want warnings like `-pedantic`, except that errors are produced rather than warnings.
Pedantic warnings as errors (-pedantic-errors)	Check this option if if you want to inhibit the display of warning messages.
Inhibit all warnings (-w)	Check this option if if you want to enable all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros.

Table continues on the next page...

Table 3-219. ARM Ltd. Windows GCC Compiler > Warnings (continued)

Option	Description
All warnings (-Wall)	Check this option if if you want to enable all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros.
Extra warnings (-Wextra)	Check this option to enable any extra warnings.
Warnings as errors (-Werror)	Check this option if if you want to make all warnings into hard errors. Source code which triggers warnings will be rejected.

3.9.6.5 ARM Ltd. Windows GCC Compiler > Miscellaneous

Use the **Miscellaneous** panel to specify compiler options.

The following table lists and describes the various options available on the **Miscellaneous** panel.

Table 3-220. ARM Ltd. Windows GCC Compiler > Miscellaneous

Option	Description
Language Standard	Select the programming language or standard to which the compiler should conform. • ISO C90 (-ansi) - Select this option to compile code written in ANSI standard C. The compiler does not enforce strict standards. For example, your code can contain some minor extensions, such as C++ style comments (//), and $ characters in identifiers. • ISO C99 (-std=c99) - Select this option to instruct the compiler to enforce stricter adherence to the ANSI/ISO standard. • Compiler Default (ISO C90 with GNU extensions) - Select this option to enforce adherence to ISO C90 with GNU extensions. • ISO C99 with GNU Extensions (-std=gnu99)
Enable Assembler Listing	Enables the assembler to create a listing file as it compiles assembly language into object code. Default: `-adhlns="$@.lst"`
Do not inline functions (-fno-inline-functions)	Check this option if you do not wnat to inline function.
char is signed (-fsigned-char)	Check this option if you want to ensure that the char is signed.
Bitfield are unsigned (-funsigned-bitfields)	Check this option to ensure bitfields are unsigned.
Verbose (-v)	Check this option if if you want the IDE to show each command-line that it passes to the shell, along with all progress, error, warning, and informational messages that the tools emit. This setting is equivalent to specifying the -v

Table continues on the next page...

Table 3-220. ARM Ltd. Windows GCC Compiler > Miscellaneous (continued)

Option	Description
	command-line option. By default this checkbox is clear. The IDE displays just error messages that the compiler emits. The IDE suppresses warning and informational messages.
Other flags	Specifies the compiler flags. The default value is: -c -fmessage-length=0

3.9.7 ARM Ltd. Windows GCC Linker

Use the **ARM Ltd. Windows GCC Linker** panel to specify the linker behaviour. The following table lists and describes the various options available on the **ARM Ltd. Windows GCC Linker** panel.

Table 3-221. ARM Ltd. Windows GCC Linker

Option	Description
Command	Shows the location of the assembler executable file.
All options	Shows the actual command line the assembler will be called with.
Expert Settings Command line pattern	Shows the expert settings command line parameters; default is `"${VSPAGCCToolsDir}/${COMMAND}" ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`
Command	Shows the location of the assembler executable file.

3.9.7.1 ARM Ltd. Windows GCC Linker > General

Use the **General** panel to specify the linker behaviour. The following table lists and describes the various options available on the **General** panel.

Table 3-222. ARM Ltd. Windows GCC Linker > General

Option	Description
Script file (-T)	This option passes the `-T` argument to the linker file
Do not use standard start files (-nostartfiles)	This option passes the `-nostartfiles` argument to the linker file. It does not allow the use of the standard start files.
Do not use default libraries (-nodefaultlibs)	This option passes the `-nodefaultlibs` argument to the linker file. It does not allow the use of the default libraries.

Table continues on the next page...

Table 3-222. ARM Ltd. Windows GCC Linker > General (continued)

Option	Description
No startup or default libs (-nostdlib)	This option passes the `-nostdlib` argument to the linker file. It does not allow the use of startup or default libs.
Remove unused sections (-Xlinker --gc-sections)	This option passes the `-Xlinker --gc-sections` argument to the linker file. It removes the unused sections.
Print removed sections (-Xlinker --print-gc-sections)	This option passes the `-Xlinker --print-gc-sections` argument to the linker file. It ptints the removed sections.
Omit all symbol information (-s)	This option passes the `-s` argument to the linker file. This option omits all symbol information.

3.9.7.2 ARM Ltd. Windows GCC Linker > Libraries

Use the **Libraries** panel to specify the libraries and their search paths if the libraries are available in nonstandard location. You can specify multiple additional libraries and library search paths. The following table lists and describes the various options available on the **Libraries** panel.

Table 3-223. ARM Ltd. Windows GCC Linker > Libraries

Option	Description
Libraries (`-l`)	This option enables the linker to search a standard list of directories for the library, which is actually a file named ' `liblibrary.a`'. The linker then uses this file as if it had been specified precisely by name. The directories searched includes several standard system directories plus any that you specify with ' `-L`'. The only difference between using an ' `-l`' option and specifying a file name is that ' `-l`' surrounds library with ' `lib`' and ' `.a`' and searches several directories.
Library search path (`-L`)	This option lists paths that the VSPA linker searches for libraries. The linker searches the paths in the order it appears in the list.

3.9.7.3 ARM Ltd. Windows GCC Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-224. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.9.7.4 ARM Ltd. Windows GCC Linker > Miscellaneous

Use the **Miscellaneous** panel to specify linker options. The following table lists and describes the various options available on the **Miscellaneous** panel.

Table 3-225. ARM Ltd. Windows GCC Linker > Miscellaneous

Option	Description
Linker flags	This option specifies the flags to be passed with the linker file.
Other objects	This option lists paths that the VSPA linker searches for objects. The linker searches the paths in the order shown in this list.
Map Filename	This option specifies the map filename. Default: `${BuildArtifactFileBaseName}.map`
Cross Reference (-Xlinker --cref)	Check this option to instruct the linker to list cross-reference information on symbols. This includes where the symbols were defined and where they were used, both inside and outside macros.
Print link map (-Xlinker --printf-map)	Check this option to instruct the linker to print the map file.
Verbose (-v)	Check this option to show verbose information, including hex dump of program segments in applications; default setting
Other flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.9.8 ARM Ltd. Windows GCC Disassembler

Use this panel to specify the command, options, and expert settings for the GCC disassembler settings. The following table lists and describes the various options available on the **GCC Disassembler** panel.

Table 3-226. ARM Ltd. Windows GCC Disassembler

Option	Description
Command	Shows the location of the GCC compiler executable file used for building the project.
All options	Shows the actual command line the compiler will be called with.
Expert settings	Shows the expert settings command line parameters; default is `"${ARMSourceryDir}/${COMMAND}" ${INPUTS} ${FLAGS}`
Command line pattern	

3.9.8.1 ARM Ltd. Windows GCC Disassembler > Disassembler Settings

Use this panel to specify the GCC disassembler settings. The following table lists and describes the various options available on the **GCC Disassembler Settings** panel.

Table 3-227. ARM Ltd. Windows GCC Disassembler > Disassembler Settings

Option	Description
Disassemble All Section Content (including debug information) (-D)	Check this option to specify the -D command to the disassembler, to disassemble all section content and sends the output to a file. This command is global and case-sensitive.
Disassemble Executable Section Content (-d)	Check this option to specify the `-d` command to the disassembler, to disassemble all executable content and send output to a file.
Intermix Source Code With Disassembly (-S)	Check this option to specify the `-S` command to the disassembler, to convert jbsr into jsr.
Display All Header Content (-x)	Check this option to specify the `-x` command to the disassembler, to display the contents of all headers.
Display Archive Header information (-a)	Check this option to specify the `-a` command to the disassembler, to display the archive header information.
Display Overall File Header content (-f)	Check this option to specify the `-f` command to the disassembler, to display the contents of the overall file header.
Display Object Format Specific File Header Contents (-p)	Check this option to specify the `-p` command to the disassembler, to display the file header contents and object format
Display Section Header Content (-h)	Check this option to specify the `-h` command to the disassembler, to display the section header of the file.

Table continues on the next page...

Table 3-227. ARM Ltd. Windows GCC Disassembler > Disassembler Settings (continued)

Option	Description
Display Full Section Content (-s)	Check this option to specify the `-s` command to the disassembler, to display the full section of the file.
Display Debug Information (-g)	Check this option to specify the `-g` command to the disassembler, to display debug information in the object file.
Display Debug Information Using ctage Style (-e)	Check this option to specify the `-e` command to the disassembler, to display debug information using the ctags style.
Display STABS Information (-G)	Check this option to specify the `-G` command to the disassembler, to display any STABS information in the file, in raw form.
Display DWARF Information (-W)	Check this option to specify the `-W` command to the disassembler, to display any DWARF information in the file.
Display Symbol Table Content (-t)	Check this option to specify the `-t` command to the disassembler, to display the contents of the symbol tables.
Display Dynamic Symbol Table Content (-T)	Check this option to specify the `-T` command to the disassembler, to display the contents of the dynamic symbol table.
Display Relocation Entries (-r)	Check this option to specify the `-r` command to the disassembler, to display the relocation entries in the file.
Display Dynamic Relocation Entries (-R)	Check this option to specify the `-R` command to the disassembler, to `display the dynamic relocation entries in the file`.

3.9.9 ARM Ltd. Windows GCC C Preprocessor

Use the **ARM Ltd. Windows GCC C Preprocessor** to specify specify the command, options, and expert settings for

the preprocessor.

The following table lists and describes the various options available on the **ARM Ltd. Windows GCC C Preprocessor** panel.

Table 3-228. Tool Settings - ARM Ltd. Windows GCC C Preprocessor

Option	Description
Command	This option shows the location of the linker executable file.
All options	This option shows the actual command line the linker will be called with.
Expert settings	This option shows the expert settings command line parameters; default is `"${ARMSourceryDir}/${COMMAND}" ${INPUTS} ${FLAGS}`
Command line pattern	

3.9.9.1 ARM Ltd. Windows GCC C Preprocessor > Preprocessor Settings

Use the **ARM Ltd. Windows GCC C Preprocessor** settings to specify specify the command, options, and expert settings for the preprocessor.

The following table lists and describes the various options available on the **ARM Ltd. Windows GCC C Preprocessor** panel.

Table 3-229. ARM Ltd. Windows GCC C Preprocessor > Preprocessor Settings

Option	Description
Handle Directives Only (fdirectives-only)	Check this option to specify the `-fdirectives-only` command to the preprocessor to handle only directives.
Print Header File Names (-H)	Check this option to specify the `-H` command to the preprocessor if you want to print header filenames.
Do not search system directories (-nostdinc)	Check this option if you do not want the assembler to search the system directories. By default, this checkbox is clear. The assembler performs a full search that includes the system directories.

3.9.9.2 ARM Ltd. Windows GCC C Preprocessor > Directories

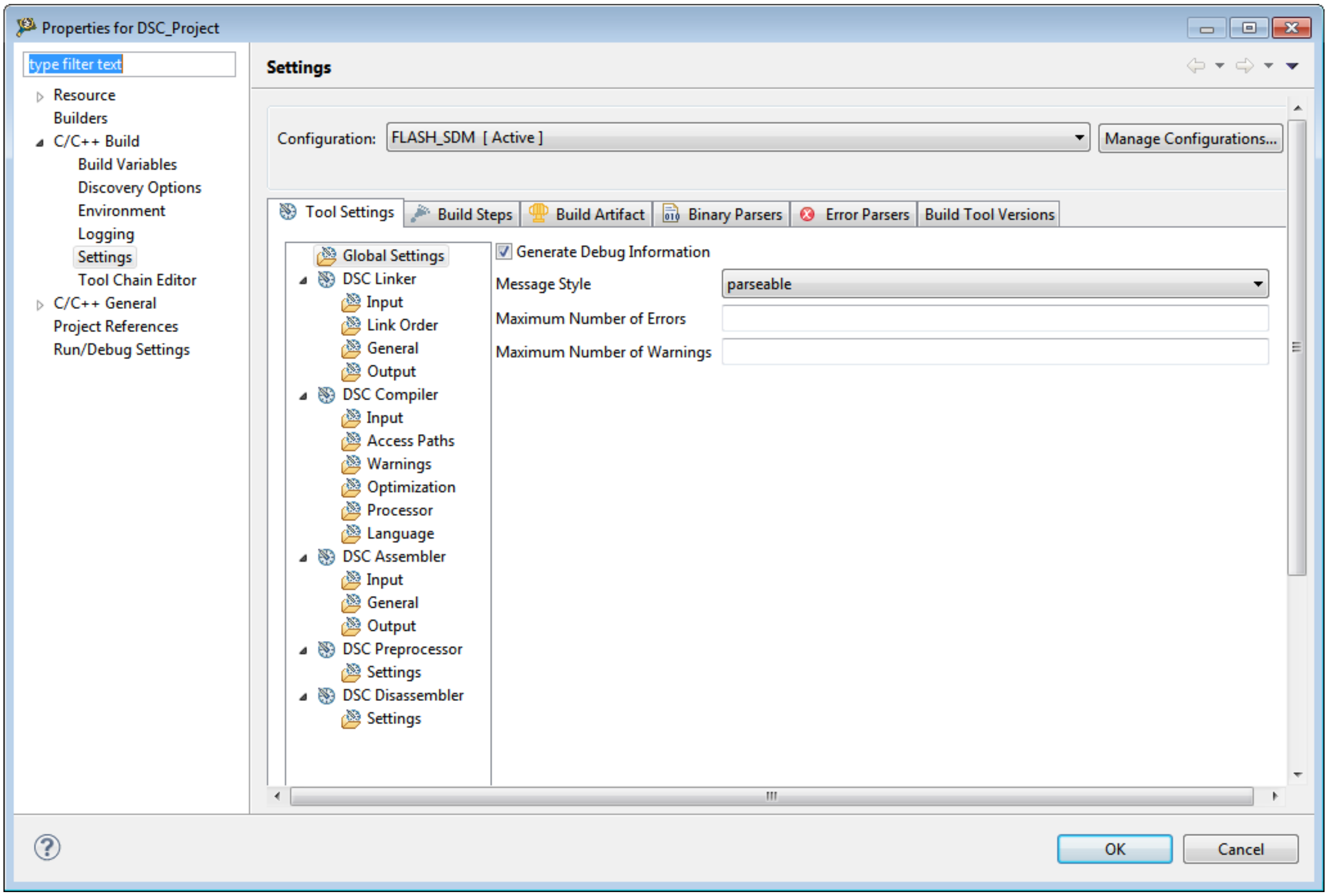
Use the **ARM Ltd. Windows GCC C PreprocessorDirectories** panel to include paths.

Table 3-230. ARM Ltd. Windows GCC C Preprocessor > Directories

Option	Description
Include User Search Paths (-i)	Enables you to add new directories to the list of directories where the user files are searched. Note: Manually edit the absolute paths if the project is moved to a new location. The path will be correct, however, for the location in which the project was originally created. Furthermore, it is a good idea to open the settings of the new project and click 'Apply' to avoid the problem of disappearing user-added include paths.

3.10 Build Properties for DSC

The **Properties for** <*project*> window shows the corresponding build properties for DSC CPU project.



Configuration: FLASH_SDM [Active]

☑ Generate Debug Information

Message Style: parseable

Maximum Number of Errors:

Maximum Number of Warnings:

Figure 3-38. Build Properties - DSC

The following table lists the build properties specific to developing software for DSC.

The properties that you specify in the **Tool Settings** panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> window.

Table 3-231. Build Properties for DSC

Build Tool	Build Properties Panels
Global Settings	Global Settings
DSC Linker	DSC Linker > Input
	DSC Linker > General
	DSC Linker > Output
DSC Compiler	DSC Compiler > Input
	DSC Compiler > Access Paths
	DSC Compiler > Warnings
	DSC Compiler > Optimization

Table continues on the next page...

Table 3-231. Build Properties for DSC (continued)

Build Tool	Build Properties Panels
	DSC Compiler > Processor
	DSC Compiler > Language
DSC Assembler	DSC Assembler > Input
	DSC Assembler > General
	DSC Assembler > Output
DSC Preprocessor	DSC Preprocessor > Settings
DSC Disassembler	DSC Disassembler > Settings

3.10.1 Global Settings

Use this panel to specify the global settings the DSC architecture uses. The build tools (compiler, linker, and assembler) then use the properties set in this panel to generate CPU-specific code.

The following table lists and describes the global settings options for DSC.

Table 3-232. Tool Settings - Global Settings

Option	Description
Generate Debug Information	Check to generate symbolic information for debugging the build target.
Message Style	List options to select message style. • **GCC (default)** - Uses the message style of the Gnu Compiler Collection tools • **MPW** - Uses the Macintosh Programmer's Workshop (MPWï¿½) message style • **standard** - Uses the standard message style • **IDE** - Uses context-free machine parseable message style • **Enterprise-IDE -** Uses CodeWarrior's Integrated Development Environment (IDE) message style. • **parseable** - Uses parseable message style.
Maximum Number of Errors	Specify the number of errors allowed until the application stops processing.
Maximum Number of Warnings	Specify the maximum number of warnings.

3.10.2 DSC Linker

Use this panel to specify the DSC linker behavior. You can specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the input, general, and output settings.

The following table lists and describes the linker options for DSC.

Table 3-233. Tools Settings > DSC Linker Options

Option	Description
Command	Shows the location of the linker executable file. You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.
Command line pattern	

3.10.2.1 DSC Linker > Input

Use this panel to specify files the DSC Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the linker input options for DSC.

Table 3-234. Tools Settings > DSC Linker > Input

Option	Description
No Standard Library	Uses the system library access paths (specified by %MWLibraries%) and add system libraries (specified by %MWLibraryFiles%) at the end of link order.
Linker Command File	Consists of three kinds of segments, which must be in this order: • A memory segment, which begins with the MEMORY{} directive. • Optional closure segments, which begin with the FORCE_ACTIVE{}, KEEP_SECTION{}, or REF_INCLUDE{} directives. • A sections segment, which begins with the SECTIONS{} directive.
Entry Point	Specifies the program starting point: the first function the debugger uses upon program start. This default function is in file Finit_MC56F824x_5x_ISR_HW_RESET. It sets up the DSC environment before code execution. Its final task is calling main().

Table continues on the next page...

Table 3-234. Tools Settings > DSC Linker > Input (continued)

Option	Description
Library Search Paths (-L)	Specifies the search pathname of libraries or other resources related to the project. Type the pathname into this text box. Alternatively, click **Workspace** or **File system** , then use the subsequent dialog box to browse to the correct location.
Library Recursive Search Paths (-lr)	Specifies the recursive user paths that the CodeWarrior IDE searches to find files in your project. You can add several kinds of paths including absolute and project relative.
Additional Libraries	Specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.
Force Active Symbols	This option allows you to specify symbols that you do not want the linker to deadstrip. You must specify the symbol(s) you want to keep before you use the `SECTIONS` keyword.

3.10.2.2 DSC Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-235. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.10.2.3 DSC Linker > General

Use this panel to specify the general linker behavior.

The following table lists and describes the linker options for DSC.

Table 3-236. Tools Settings > DSC Linker > General

Option	Description
Dead-Strip Unused Code	Determines whether to pool constants from all functions in a file.

Table continues on the next page...

Table 3-236. Tools Settings > DSC Linker > General (continued)

Option	Description
Suppress Link Warnings	Prevents the IDE from displaying linker warning messages.
Large Data Memory Model	The large data memory model allows data to be placed in memory at addresses greater than the 16-bit address limitation of the small data model. When selected this option informs the compiler that global and static data should be addressed with the 24-bit variants of the absolute addressing modes of the device. Also in the large memory model, pointers are treated as 24-bit quantities when moved from register to register, memory to register, or register to memory.
Generates elf file for 56800EX core	Check to generate an ELF file (the default output file format) and an S-record output file for your application.
Other Flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.10.2.4 DSC Linker > Output

Use this panel to specify the output settings for the DSC linker output.

The following table lists and describes the linker options for DSC.

Table 3-237. Tools Settings > DSC Linker > Output

Option	Description
Output Type	Select output type as Application (default) or Library.
Generate Link Map	Check to generate link map.
List Unused Symbols in Map	Check to list unused symbols; appears grayed out if the Generate Link Map checkbox is not checked.
Show Transitive Closure in Map	Check show transitive closure; appears grayed out if the Generate Link Map checkbox is not checked.
Annotate Byte Symbols in Map	Checked - Linker includes B annotation for byte data types (e.g., char) in the Linker Command File. Clear - By default, the Linker does not include the B annotation in the Linker Command File. Everything without the B annotation is a word address.
Generate ELF Symbol Table	Check to generated the ELF symbol table.
Generate S-Record File	Check to generate a S-record file.
Sort by Address	Check to sort by address.
Generate Byte Addresses	Check to generate byte address.
Max S-Record Length	Specify the maximum length for S-record; appears grayed out if the Generate S-Record File checkbox is not checked. The default value is 252.
S_Record EOL Character	Specify the end-of-line character; appears grayed out if the Generate S-Record File checkbox is not checked. The default value is DOS (\r\n).

3.10.3 DSC Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the DSC Compiler tree control includes the general and the file search path settings.

The following table lists and describes the linker options for DSC.

Table 3-238. Tools Settings > DSC Compiler

Option	Description
Command	Shows the location of the compiler executable file. You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the compiler will be called with.
Expert settings	Shows the command line pattern; default is `${COMMAND} -c ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.
Command line pattern	

3.10.3.1 DSC Compiler > Input

Use this panel to specify additional files the DSC Compiler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the compiler inputs options for DSC.

Table 3-239. Tools Settings > DSC Compiler > Input

Option	Description
Prefix File	Specifies a file to be included at the beginning of every assembly file of the project. Lets you include common definitions without using an include directive in every file.
Source File Encoding	Allows you to specify the default encoding of source files. Multibyte and Unicode source text is supported. The options available are: • ASCII (default) • Auto-Detect (multibyte encoding) • System (use system locale) • UTF-8 • Shift-JIS

Table continues on the next page...

Table 3-239. Tools Settings > DSC Compiler > Input (continued)

Option	Description
	• EUC-JP • ISO-2022-JP
Allow Macro Redefinition	Enables to redefine the macros with the #define directive without first undefining them with the #undef directive.
Defined Macros	Lists the defined command-line macros.
Undefined Macros	Lists the undefined command-line macros.

3.10.3.2 DSC Compiler > Access Paths

Use this panel to specify the access paths. Access paths are directory paths the CodeWarrior tools use to search for libraries, runtime support files, and other object files.

The following table lists and describes the compiler access paths for DSC.

Table 3-240. Tools Settings > DSC Compiler > Access Paths

Option	Description
Search User Paths (#include "...")	Lets you add/update the user paths that the CodeWarrior IDE searches to find files in your project. You can add several kinds of paths including absolute and project-relative.
Search User Paths Recursively	Lets you add/update the recursive user paths that the CodeWarrior IDE searches to find files in your project. You can add several kinds of paths including absolute and project-relative.
Search System Paths (#include <...>)	Lets you add/update the system paths that the CodeWarrior IDE searches to find files in your project. You can add several kinds of paths including absolute and project-relative.
Search System Paths Recursively	Lets you add/update the recursive system paths that the CodeWarrior IDE searches to find files in your project. You can add several kinds of paths including absolute and project-relative.

3.10.3.3 DSC Compiler > Warnings

Use this panel to control how the DSC compiler formats the listing file, error and warning messages.

The following table lists and describes the compiler warnings options for DSC.

Table 3-241. Tool Settings - DSC Compiler > Warnings

Option	Description
Treat All Warnings As Errors	Check to treat all warnings as errors. The compiler will stop if it generates a warning message.
Enable Warnings	Select the level of warnings you want reported from the compiler. Custom lets you to select individual warnings. Other settings select a pre-defined set of warnings.
Illegal #pragmas (most)	Check to notify the presence of illegal pragmas.
Possible Unwanted Effects (most)	Check to notify most of the possible errors.
Extended Error Checks (most)	Check if you want to do an extended error checking.
Hidden Virtual Functions (most)	Check to generate a warning message if you declare a non-virtual member function that prevents a virtual function, that was defined in a superclass, from being called and is equivalent to pragma `warn_hidevirtual` and the command-line option `-warnings hidevirtual`.
Implicit Arithmetic Conversions (all)	Check to warn of implict arithmetic conversions.
Implicit Signed/Unsigned Conversions (all)	Check to enable warning of implict conversions between signed and unsigned variables.
Implicit Float to Integer Conversions (all)	Check to warn of implict conversions of a floating-point variable to integer type.
Implicit Integer to Float Conversions (all)	Check to warn of implict conversion of an integer variable to floating-point type.
Pointer/Integer Conversions (most)	Check to enable warnings of conversions between pointer and integers.
Unused Arguments (most)	Check to warn of unused arguments in a function.
Unused Variables (most)	Check to warn of unused variables in the code.
Unused Result from Non-Void-Returning Function (full)	Check to warn of unused result from non-void-returning functions.
Missing 'return' Value in Non-Void-Returning Function (most)	Check to warn of when a function lacks a return statement.
Expression has no Side Effect (most)	Check to issue a warning message if a source statement does not change the program's state. This is equivalent to the pragma `warn_no_side_effect`, and the command-line option `-warnings unusedexpr`.
Extra Commas (most)	Check to issue a warning message if a list in an enumeration terminates with a comma. The compiler ignores terminating commas in enumerations when compiling source code that conforms to the ISO/IEC 9899-1999 ("C99") standard and is equivalent to pragma `warn_extracomma` and the command-line `option -warnings extracomma`.
Empty Declarations (most)	Check to warn of empty declarations.
Inconsistent 'class'/'struct' Usage (most)	Check to warn of inconsistent usage of class or struct.
Incorrect Capitalization in #include "..." (most)	Check to issue a warning message if the name of the file specified in a #include "file" directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps` and the command-line option `-warnings filecaps`.

Table continues on the next page...

Table 3-241. Tool Settings - DSC Compiler > Warnings (continued)

Option	Description
Incorrect Capitalization in System #include <...> (most)	Check to issue a warning message if the name of the file specified in a #include <file> directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps_system` and the command-line option `-warnings sysfilecaps.`
Pad Bytes Added (full)	Check to issue a warning message when the compiler adjusts the alignment of components in a data structure and is equivalent to pragma `warn_padding` and the command-line option `-warnings padding.`
Undefined Macro in #if/#elif (full)	Check to issues a warning message if an undefined macro appears in #if and #elif directives and is equivalent to pragma `warn_undefmacro` and the command-line option `-warnings undefmacro.`
Non-Inlined Functions (full)	Check to issue a warning message if a call to a function defined with the inline, __inline__, or __inline keywords could not be replaced with the function body and is equivalent to pragma `warn_notinlined` and the command-line option `-warnings notinlined.`
Token not Formed by ## Operator (most)	Check to enable warnings for the illegal uses of the preprocessor's token concatenation operator (##). It is equivalent to the pragma `warn_illtokenpasting on`.

3.10.3.4 DSC Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the compiler optimization options for DSC.

Table 3-242. Tool Settings - DSC Compiler > Optimization

Option	Description
Optimization Level (-opt)	Specify the optimizations that you want the compiler to apply to the generated object code: 0-Disable optimizations. This setting is equivalent to specifying the `-O0` command-line option. The compiler generates unoptimized, linear assembly-language code. 1-The compiler performs all target-independent (that is, non-parallelized) optimizations, such as function inlining. This setting is equivalent to specifying the `-O1` command-line option. The compiler omits all target-specific optimizations and generates linear assembly-language code. 2-The compiler performs all optimizations (both target-independent and target-specific). This setting is equivalent to specifying the `-O2` command-line option. The

Table continues on the next page...

Table 3-242. Tool Settings - DSC Compiler > Optimization (continued)

Option	Description
	compiler outputs optimized, non-linear, parallelized assembly-language code. 3-The compiler performs all the level 2 optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the `-O3` command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level 2 optimizations. 4- The compiler performs all the level 3 optimizations. This setting is equivalent to specifying the `-O4 command-line option.` At this level, the compiler adds repeated subexpression elimination and loop-invariant code motion.
Speed Vs Size	Use to specify an Optimization Level greater than 0. • Speed-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a faster execution speed, as opposed to a smaller executable code size. • Size-The compiler optimizes object code at the specified Optimization Level such that the resulting binary file has a smaller executable code size, as opposed to a faster execution speed. This setting is equivalent to specifying the `-Os` command-line option.
Inter-Procedural Analysis	Control whether the compiler views single or multiple source files at compile time. • Off-Compiler compiles one file at a time. The functions are displayed in order as they appear in the source file. An object file is created for each source. • File-The compiler sees all the functions and data in a translation unit (source file) before code or data is generated. This allows inlining of functions that may not have been possible in -ipa off mode. • Program-The compiler sees all the source files of a project before code and data are generated. This allows for cross-module optimizations, including inlining. • Program-Final-The compiler sees all the source files of a project before code and data are generated. The object files are passed explicitly to the linker.
Inline Level	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. Using the -Oi=c0 option switches off inlining. Functions marked with the #pragma INLINE are still inlined. To disable inlining, use the -Oi=OFF option. • Smart- Inlines function declared with the inline qualifier • Off- No functions are inlined.
Auto Inline	Inlines small function even if they are not declared with the inline qualifier
Bottom-up Inlining	Check to control the bottom-up function inlining method. When active, the compiler inlines function code starting with the last function in the chain of functions calls, to the first one.

3.10.3.5 DSC Compiler > Processor

Use this panel to specify processor behavior. You can specify the file paths and define macros.

The following table lists and describes the compiler procesor options for DSC.

Table 3-243. Tool Settings - DSC Compiler > Processor Options

Option	Description
Hardware DO Loops	Specifies the level of hardware DO loops: No DO Loops - Compiler does not generate any No Nested DO Loops - Compiler generates hardware DO loops, but does not nest them Nested DO Loops - Compiler generates hardware DO loops, nesting them two deep. If hardware DO loops are enabled, debugging will be inconsistent about stepping into loops. Test immediately after this table contains additional Do-loop information.
Small Program Model	Checked - Compiler generates a more efficient switch table, provided that code fits into the range 0x0-0xFFFF Clear - Compilr generates an ordinary switch table. Do not check this checkbox unless the entire program code fits into the 0x0-0xFFFF memory range.
Large Data Memory Model	Checked - Extends DSP56800E addressing range by providing 24-bit address capability to instructions Clear - Does not extend address range 24-bit address modes allow access beyond the 64K-byte boundary of 16-bit addressing.
Globals Live in Lower Memory	Checked - Compiler uses 24-bit addressing for pointer and stack operations, 16-bit addressing for access to global and static data. Clear - Compiler uses 24-bit addressing for all data access. This checkbox is available only if the Large Data Model checkbox is checked.
Zero-Initialized Globals LIve in Data Instead of BSS	Checked - Globals initialized to zero reside in the .data section. Clear - Globals initialized to zero reside in the .bss section.
Segregate Data Section	Check to segregate data section.
Pad Pipeline for Debugger	Checked - Mandatory for using the debugger. Inserts NOPs after certain branch instructions to make breakpoints work reliably. Clear - Does not insert such NOPs. If you select this option, you should select the same option in the assembler panel. Selecting this option increases code size by 5 percent. But not selecting this option risks nonrecovery after the debugger comes to breakpoint branch instructions.
Generate Code for Profiling	Checked - Compiler generates code for profiling. Clear - Compiler does not generate code for profiling.
Generate elf file for 56800EX core	Checked - Compiler generates elf file for 56800EX core. Clear - Compiler does not generates elf file for 56800EX core.
Check Inline Assembly for Pipeline	Specifies pipeline conflict detection during compiling of inline assembly source code: Not Detected - compiler does not check for conflicts Conflict Error - compiler issues error

Table continues on the next page...

Table 3-243. Tool Settings - DSC Compiler > Processor Options (continued)

Option	Description
	messages if it detects conflicts Conflict Error/Hardware Stall Warning - compiler issues error messages if it detects conflicts, warnings if it detects hardware stalls
Check C Source for Pipeline	Specifies pipeline conflict detection during compiling of C source code: Not Detected - compiler does not check for conflicts Conflict error - compiler issues error messages if it detects conflicts

3.10.3.6 DSC Compiler > Language

Use this panel direct the DSC compiler to apply specific processing modes to the language source code. You can compile source files with just one collection at a time. To compile source files with multiple collections, you must compile the source code sequentially. After each compile iteration change the collection of settings that the DSC compiler uses.

The following table lists and describes the compiler optimization options for DSC.

Table 3-244. Tool Settings - DSC Compiler > Language Settings

Option	Description
ANSI Strict	Check to enable C compiler operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ ISO specification to all input files. This setting is equivalent to specifying the `- ansi` command-line option. The compiler issues a warning for each ANSI/ISO extension it finds.
ANSI Keywords Only	**Check to** generate an error message for all non-standard keywords (ISO/IEC 9899-1990 C, ï¿½6.4.1). If you must write source code that strictly adheres to the ISO standard, enable this setting; is equivalent to pragma `only_std_keywords` and the command-line option `-stdkeywords`.
Enums Always Int	**Check to u** se signed integers to represent enumerated constants and is equivalent to pragma `enumsalwaysint` and the command-line option `-enum`.
Use Unsigned Chars	**Check to t** reat char declarations as unsigned char declarations and is equivalent to pragma `unsigned_char` and the command-line option `-char unsigned.`
Require Function Prototypes	Check to enforce the requirement of function prototypes. The compiler generates an error message if you define a previously referenced function that does not have a prototype. If you define the function before it is referenced but do not give it a prototype, this setting causes the compiler to issue a warning message.

Table continues on the next page...

Table 3-244. Tool Settings - DSC Compiler > Language Settings (continued)

Option	Description
Expand Trigraphs	**Check to** recognize trigraph sequences (ISO/IEC 9899-1990 C, ï¿½5.2.1.1); is equivalent to **pragma** `trigraphs` **and the command-line option** `-trigraphs.`
Legacy for-scoping	**Check to g** enerate an error message when the compiler encounters a variable scope usage that the ISO/IEC 14882-1998 C++ standard disallows, but is allowed in the C++ language specified in The Annotated C++ Reference Manual.
Reuse Strings	**Check to** store only one copy of identical string literals and is equivalent to opposite of the pragma `dont_reuse_strings` and the command-line `option -string reuse.`
Pool Strings	**Check to** collect all string constants into a single data section in the object code it generates and is equivalent to pragma `pool_strings` and the command-line `option -strings pool.`
Other flags	Specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI. **Note** : To enable CodeWarrior MCU V10.x to generate .lst file for each source file in DSC you need to specify -S in the **Other Flags** option.

3.10.4 DSC Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler. Additionally, the Assembler tree control includes the general and include file search path settings.

The following table lists and describes the compiler optimization options for DSC.

Table 3-245. Tool Settings - DSC Assembler

Option	Description
Command	Shows the location of the assembler executable file. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}.`
Command line pattern	

3.10.4.1 DSC Assembler > Input

Use this panel to specify additional files the DSC Assembler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the compiler optimization options for DSC.

Table 3-246. Tool Settings - DSC Assembler > Input

Option	Description
Prefix File	Specify a prefix file that you want the compiler to include at the top of each file.
Always Search User Paths (-nosyspath)	Performs a search of the user paths, treating #include statements of the form #include <xyz> the same as the form #include " `xyz"`.
User Path (-i)	Enables you to add new directories to the list of directories where the user files are searched.
User Recursive Path (-ir)	Enables you to add new directories to the list of directories recursively where the user files are searched.
System Path (-I- -I)	Enables you to add new directories to the list of directories where the system files are searched.
System Recursive Path (-I- -ir)	Enables you to add new directories to the list of directories recursively where the system files are searched.

3.10.4.2 DSC Assembler > General

Use this panel to specify additional files the DSC Assembler should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the assembler options for DSC.

Table 3-247. Tool Settings - DSC Assembler > General

Option	Description
Identifiers are Case-Sensitive	Clear to instruct the assembler to ignore case in identifiers. By default, the option is checked.

Table continues on the next page...

Table 3-247. Tool Settings - DSC Assembler > General (continued)

Option	Description
Assert NOPs on Pipeline Conflicts	Checked - Assembler automatically resolves pipeline conflicts by inserting NOPs. Clear - Assembler does not insert NOPs; it reports pipeline conflicts in error messages. NOP is optional. The core will stall for you (delay the required time) even if you do not put the NOP.
Emit Warnings for NOP Assertions	Checked - Assembler issues a warning any time it inserts a NOP to prevent a pipeline conflict. Clear - Assembler does not issue such warnings. This checkbox is available only if the Assert NOPs on pipeline conflicts checkbox is checked.
Emit Warnings for Hardware Stalls	Checked - Assembler warns when a hardware stall occurs upon execution. Clear - Assembler does not issue such warnings. This option helps optimize the cycle count.
Pad Pipeline for Debugger	Checked - Mandatory for using the debugger. Inserts NOPs after certain branch instructions to make breakpoints work reliably. Clear - Does not insert such NOPs. If you select this option, you should select the same option in the processor settings panel. Selecting this option increases code size by 5 percent. But not selecting this option risks nonrecovery after the debugger comes to breakpoint branch instructions.
Emit Warnings for Odd SP Increment/Decrement	Checked - Enables assembler warnings about instructions that could misalign the stack frame. Clear - Does not enable such warnings.
Allow Legacy Instructions (default to 16-bit memory models)	Checked - Assembler permits legacy DSP56800 instruction syntax. Clear - Assembler does not permit this legacy syntax. Selecting this option sets the Default Data Memory Model and Default Program Memory Model values to 16 bits.
Generates elf file for 56800EX core	Check to generate an ELF file (the default output file format) and an S-record output file for your application.
Default Data Memory Model	Specifies 16 or 24 bits as the default size. Factory setting: 16 bits.
Default Program Memory Model	Specifies 16, 19, or 21 bits as the default size. Factory setting: 19 bits.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI. **Note:** To enable CodeWarrior MCU V10.x to generate .lst file for each source file in DSC, you need to specify -S in the Other Flags option.

3.10.4.3 DSC Assembler > Output

Use this panel to control how the assembler generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

The following table lists and describes the assembler output options for DSC.

Table 3-248. Tool Settings - DSC Assembler > Output

Option	Description
Generate Listing File	Instructs the assembler to generate a disassembly output file. The disassembly output file contains the file source, along with line numbers, relocation information, and macro expansion.
Expand Macros in Listing File	Checked - Assembler macros expand in the assembler listing. Clear - Assembler macros do not expand. This checkbox is available only if the Generate Listing File checkbox is checked.

3.10.5 DSC Preprocessor

Use this panel to specify the preprocessor settings for DSC.

The following table lists and describes the preprocessor options for DSC.

Table 3-249. Tool Settings - DSC Preprocesor

Option	Description
Command	Shows the location of the assembler executable file. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} -E ${FLAGS} ${INPUTS}`.
Command line pattern	

3.10.5.1 DSC Preprocessor > Settings

Use this panel to specify the preprocessor settings of DSC Preprocessor.

The following table lists and describes the preprocessor settings options for DSC.

Table 3-250. Tool Settings - DSC Preprocessor > Settings

Option	Description
Emit File/Line Breaks	Check to notify file breaks (or #line breaks) appear in the output.
Emit #pragma directives	Check to show pragma directives in the preprocessor output. Essential for producing reproducible test cases for bug reports.
Emit #line Directives	Check to display file changes in comments (as before) or in #line directives.
Show Full Path	Check to control whether file changes show the full path or the base filename of the file.
Keep Comments	Check to display comments in the preprocessor output.
Keep Whitespace	Check to copy whitespaces in preprocessor output. This is useful for keeping the starting column aligned with the original source, though the compiler attempts to preserve space within the line. This does not apply when macros are expanded.

3.10.6 DSC Disassembler

Use this panel to specify the command, options, and expert settings for DSC Disassembler.

The following table lists and describes the disassembler options for DSC.

Table 3-251. Tool Settings - DSC Disassembler

Option	Description
Command	Shows the location of the preprocessor executable file. Default value is `""${DSC_ToolsDir}/mwld56800e"`. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert settings	Shows the command line pattern; default is `${COMMAND} -dis ${FLAGS} ${INPUTS}`
Command line pattern	

3.10.6.1 DSC Disassembler > Settings

Use this panel to specify disassembler settings.

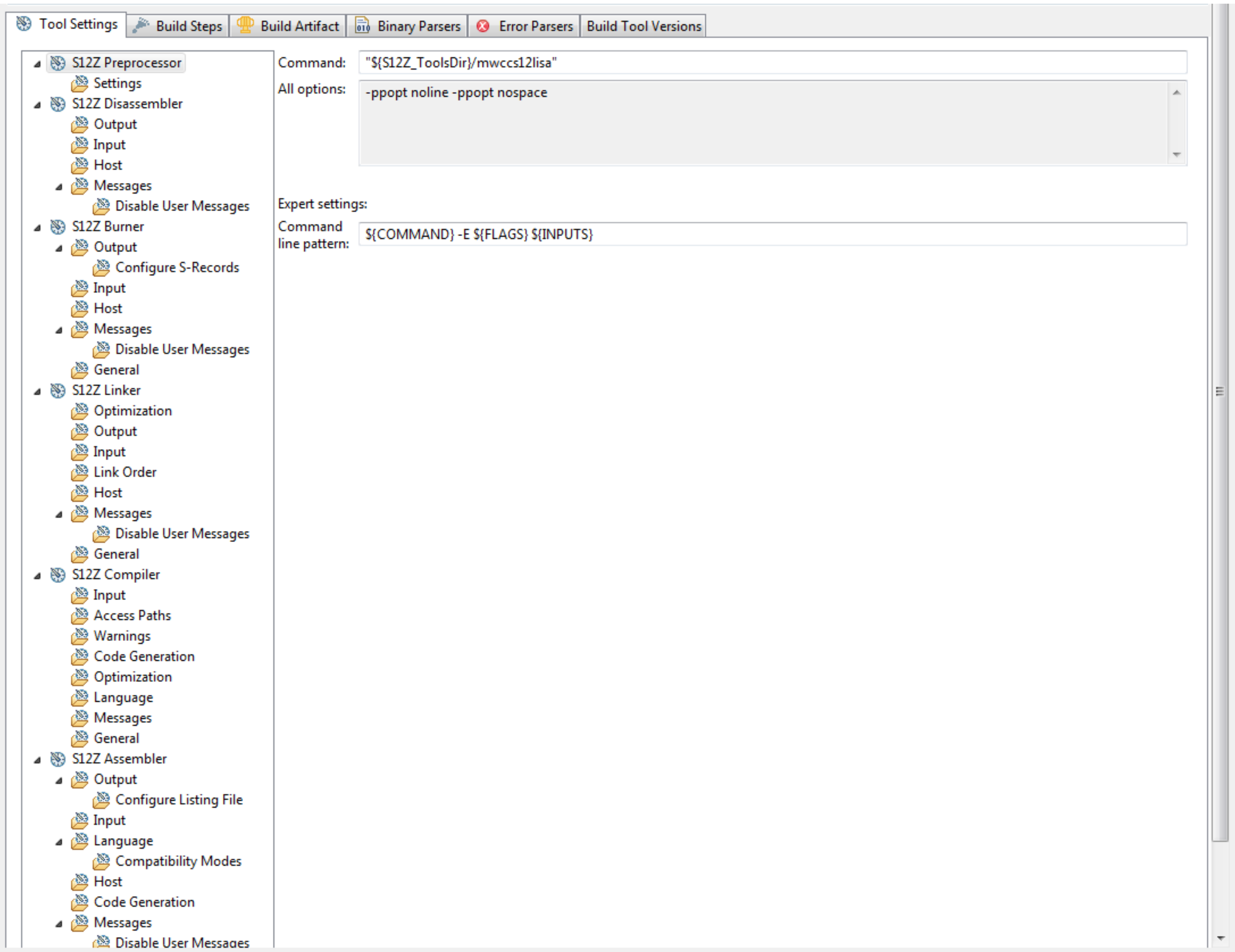
The following table lists and describes the disassembler settings options for DSC.

Table 3-252. Tool Settings - DSC Disassembler > Settings

Option	Description
Show Headers	Check to display headers in the listing file; disassembler writes listing headers, titles, and subtitles to the listing file
Show Symbol and String Tables	Check to display symbol and string tables directives to the listing file
Verbose Information	Tells the compiler to provide verbose, cumulative information in messages.
Show Relocations	Check to have the disassembler show information about relocated symbols. Clear to prevent the disassembler from showing information about relocated symbols.
Show Code Modules	Check to show core modules in the listing file
Show Extended Mnemonics	Check to show the extended mnemonics in the listing file
Show Addresses and Opcodes	Check to show the addresses and object code in the listing file
Show Source Code	Check to show the source code in the listing file
Show Comments	Check to show the comments in the listing file
Show Data Modules	Check to show the data modules in the listing file
Show Exception Tables	Check to disassemble exception tables in the listing file
Show Debug Information	Check to generate symbolic information for debugging the build target

3.11 Build Properties for S12Z

The **Properties for** <*project*> dialog box shows the corresponding build properties for S12Z CPU project.



Command: "${S12Z_ToolsDir}/mwccs12lisa"

All options: -ppopt noline -ppopt nospace

Expert settings:

Command line pattern: ${COMMAND} -E ${FLAGS} ${INPUTS}

Figure 3-39. Build Properties - S12Z

The following table lists the build properties specific to developing software for S12Z.

The properties that you specify in the **Tool Settings** panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> dialog box.

Table 3-253. Build Properties for S12Z

Build Tool	Build Properties Panels
S12Z Preprocessor	S12Z Preprocessor > Settings
S12Z Disassembler	S12Z Disassembler > Output
	S12Z Disassembler > Input
	S12Z Disassembler > Host
	S12Z Disassembler > Messages
	S12Z Disassembler > Messages > Disable User Messages
S12Z Burner	S12Z Burner > Output > Configure S-Record
	S12Z Burner > Input

Table continues on the next page...

Table 3-253. Build Properties for S12Z (continued)

Build Tool	Build Properties Panels
	S12Z Burner > Host
	S12Z Burner > Messages
	S12Z Burner > Messages > Disable User Messages
	S12Z Burner > General
S12Z Linker	S12Z Linker > Optimization
	S12Z Linker > Output
	S12Z Linker > Input
	S12Z Linker > Host
	S12Z Linker > Messages
	S12Z Linker > Messages > Disable User Messages
	S12Z Linker > General
S12Z Compiler	S12Z Compiler > Input
	S12Z Compiler > Access Paths
	S12Z Compiler > Warnings
	S12Z Compiler > Code Generation
	S12Z Compiler > Optimization
	S12Z Compiler > Language
	S12Z Compiler > Messages
	S12Z Compiler > General
S12Z Assembler	S12Z Assembler > Output
	S12Z Assembler > Output > Configure Listing File
	S12Z Assembler > Input
	S12Z Assembler > Language
	S12Z Assembler > Language > Compatibility modes
	S12Z Assembler > Host
	S12Z Assembler > Code Generation
	S12Z Assembler > Messages
	S12Z Assembler > Messages > Disable User Messages
	S12Z Assembler > General

3.11.1 S12Z Burner

Use the Burner for S12Z Preference Panel to select the settings for the S12Z Burner.

The following table lists and describes the burner options for S12Z.

Table 3-254. Tool Settings - S12Z Burner Options

Option	Description
Command	Shows the location of the burner executable file. Default value is: `"${S12Z_ToolsDir}/burner"`. You can specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the burner will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${INPUTS}`.
Command line pattern	

3.11.1.1 S12Z Burner > Output > Configure S-Record

Use this panel to control how the burner generates the output file, as well as error and warning messages. You can specify whether to allocate constant objects in ROM, generate debugging information, and strip file path information.

Use this panel to configure the S-record options of the Burner.

The following table lists and describes the general options for **S12Z configure S-Record** panel.

Table 3-255. Tool Settings - S12Z Burner > Output > Configure S-Record Options

Option	Description
Select all	This option disables the generation of all records, from the start (S0) to the end records (S7, S8, or S9).
No path in S0-record	This option removes the path (if present) from the file name in the S0 record.
No S9-record	This option disables the generation of S9-record.
No S8-record	This option disables the generation of S8-record.
No S7-record	This option disables the generation of S7-record.
No S0-record	This option disables the generation of S0-record.

3.11.1.2 S12Z Burner > Input

Use this panel to specify the execute command file of the S12Z Burner input.

The following table lists and describes the inputs options for burner.

Table 3-256. Tool Settings - S12Z Burner > Input Options

Option	Description
Execute command file	This option allows you to execute a Batch Burner command file.

3.11.1.3 S12Z Burner > Host

Use this panel to specify the host settings of the S12Z.

The following table lists and describes the memory model options for S12Z.

Table 3-257. Tool Settings - Host

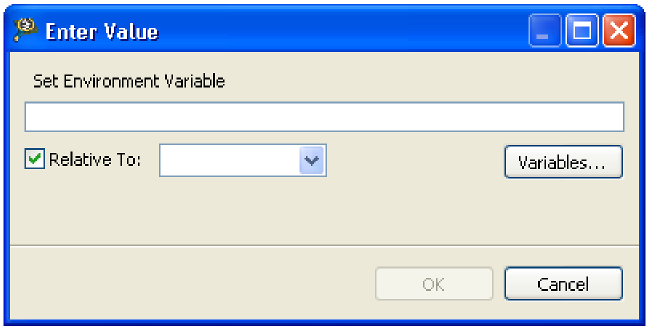
Option	Description
Borrow License Feature	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait Until a License is Available from Floating License Server	By default, if a license is not available from the floating license server, then the application will immediately return. With `-LicWait` set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	This option allows you to select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the `View` option. • `-ViewWindow`, the application appears with its normal window. • `-ViewMin` the application appears as an icon in the task bar. • `-ViewMax`, the application appears maximized (filling the whole screen). • `-ViewHidden`, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).
Set Environment Variable	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.

The following table lists and describes the toolbar buttons for the **Set Environment Variable** option.

Table 3-258. Toolbar Buttons - Set Environment Variable Option

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Environment Variable** option in the **S12Z Burner > Host** panel.

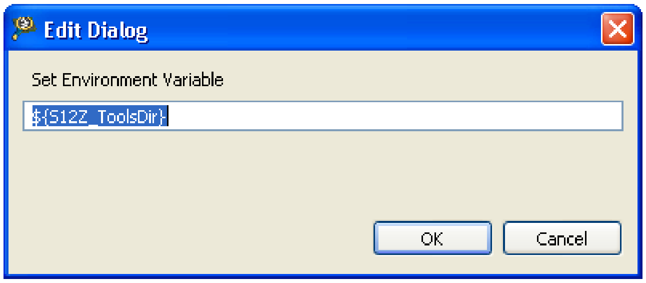

Enter Value

Set Environment Variable

☑ Relative To:

Figure 3-40. Set Environment Variable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Environment Variable** option in the **S12Z Burner > Host** panel.


Edit Dialog

Set Environment Variable

${S12Z_ToolsDir}

Figure 3-41. Set Environment Variable - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.1.4 S12Z Burner > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-259. Tool Settings - S12Z Burner > Messages Options

Option	Description
Don't Print INFORMATION Messages	This option allows you to disable the INFORMATION messsges. The `-W1` command inhibits the INFORMATION message reporting.
Don't Print INFORMATION or WARNING Messages	This option allows you to disable the printing of INFORMATION or WARNING messages. The `-W2` command suppresses all messages of the type INFORMATION or WARNING.
Create err.log Error File	You can use this option to enable the burner to create the err.log error file. The `-WErrFileOn` command creates or deletes the `err.log` file when the application is finished. When the errors occur, 16-bit window environments use the `err.log` files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the `err.log` file. By default, this checkbox is checked.
Create Error Listing File	You can use this option to create an error listing file. The `-WOutFileOn` command creates an error listing file. This option controls whether an error listing file should be created. The error listing file contains a list of all messages and errors that are created during processing. By default, this checkbox is checked.
Cut File Names in Microsoft Format to 8.3	This option truncates the filename to the 8.3 format. The `-Wmsg8x3` command cuts the filenames in Microsoft Format to 8.3. Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT.
Set Message File Format for Batch Mode	Use this option to set the message file format for batch mode. The `-WmsgFb(-WmsgFbi,-WmsgFbm)` command sets the message file format for the batch mode. This option starts the Compiler with additional arguments (for example, files and Compiler options). If you start the Compiler with the arguments (for example, from the Make Tool or with the appropriate argument from an external editor), the Compiler

Table continues on the next page...

Table 3-259. Tool Settings - S12Z Burner > Messages Options (continued)

Option	Description
	compiles the files in a batch mode. No Compiler window is visible and the Compiler terminates after the job completion. The options available are: • Verbose Format • Microsoft Format (default)
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for No File Info (e.g. %K %d: %m\n)	Use this option to set the message format for no file information. If there is no file information available for a message, then the `<string>` in the `-WmsgFonf<string>` command defines the message format string to use. The supported formats are: • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, if the path or the extension contains a space • `%'`: A ' if the filename, the path or the extension contains a space
Message Format for No Position Info (e.g. %"%f%e%": %K %d: %m\n)	This option allows you to set the message format for no position information. If there is no position information available for a message, then the `<string>` in the `-WmsgFonp<string>` command defines the message format string to use. The supported formats are: • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text)

Table continues on the next page...

Table 3-259. Tool Settings - S12Z Burner > Messages Options (continued)

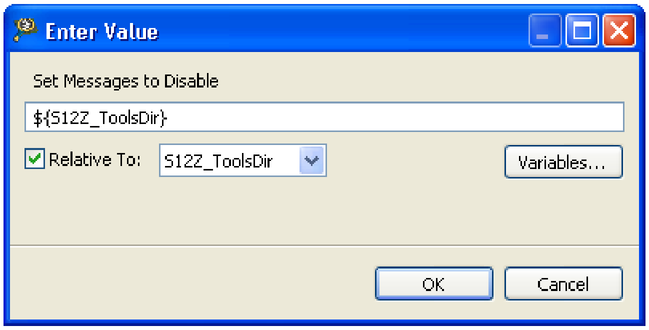
Option	Description
	- `%%`: Percent (example, %) - `\n`: New line - `%"`: A " if the filename, if the path or the extension contains a space - `%'`: A ' if the filename, the path or the extension contains a space
Maximum Number of Error Messages	This option allows you to set the maximum number of error messages to be displayed. The `<number>` in the `-WmsgNe<number>` command sets the number of error messages to be displayed.
Maximum Number of Information Messages	This option allows you to set the amount of information messages that are logged. The `<number>` in the `-WmsgNi<number>` command specifies the maximum number of information messages allowed.
Set Messages to Disable	This option allows you to disable the specified messages. The `-WmsgSd<number>` command sets a message to disable, where `<number>` is the message number to be disabled, e.g., 1801.
Set Messages to Error	This option changes a message to an error message. The argument `<number>` in the command `-WmsgSe<number>` sets the specified message number to be an error, e.g., 1853.
Set Messages to Warning	This option sets a message to a warning message. The argument `<number>` of the `-WmsgSw<number>` command, sets the specified error number to be a warning, e.g., 2901.
Set Messages to Information	This option sets a message to an information message. The argument `<number>` of the command `-WmsgSi<number>` sets the specified message number to be an information, e.g., 1853.

The following table lists and describes the toolbar buttons for the **Set Messages to Disable** , **Set Messages to Error** , **Set Messages to Warning** and **Set Messages to Information** options of the **S12Z Burner > Messages** panel.

Table 3-260. Search Paths Toolbar Buttons - Messages Panel

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Messages to Disable** option in the **S12Z Burner > Messages** panel.



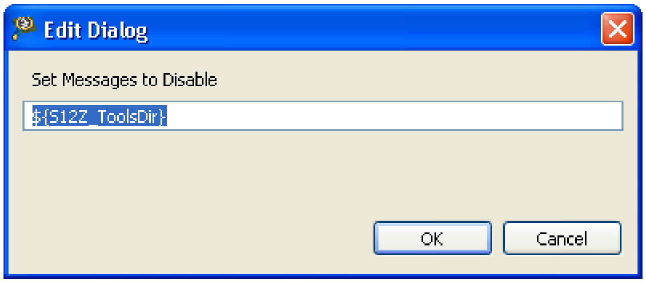
Set Messages to Disable

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-42. Set Messages to Disable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Disable** option in the **S12Z Burner > Messages** panel.

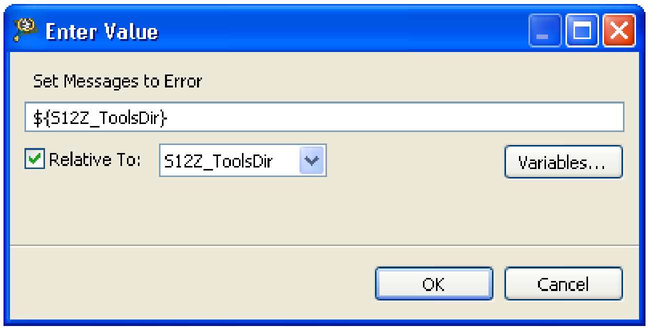


Set Messages to Disable

${S12Z_ToolsDir}

Figure 3-43. Set Messages to Disable - Edit Dialog

The following figur shows the **Enter Value** dialog box for the **Set Messages to Error** option in the **S12Z Burner > Messages** panel.



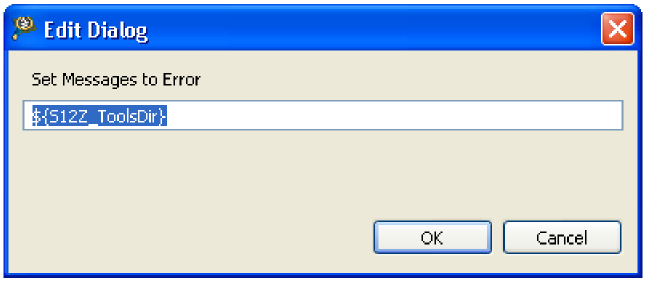
Set Messages to Error

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-44. Set Messages to Error - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Error** option in the **S12Z Burner > Messages** panel.



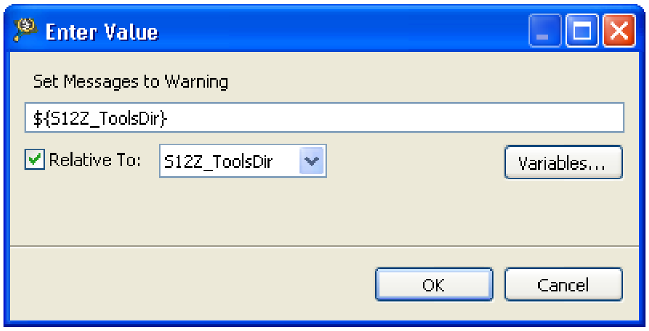
Edit Dialog

Set Messages to Error

${S12Z_ToolsDir}

Figure 3-45. Set Messages to Error - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Warning** option in the **S12Z Burner > Messages** panel.



Enter Value

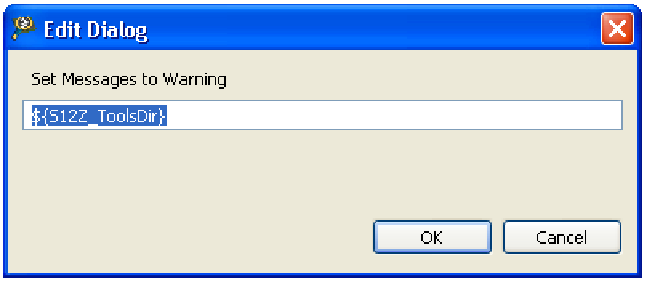
Set Messages to Warning

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-46. Set Messages to Warning - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Warning** option in the **S12Z Burner > Messages** panel.



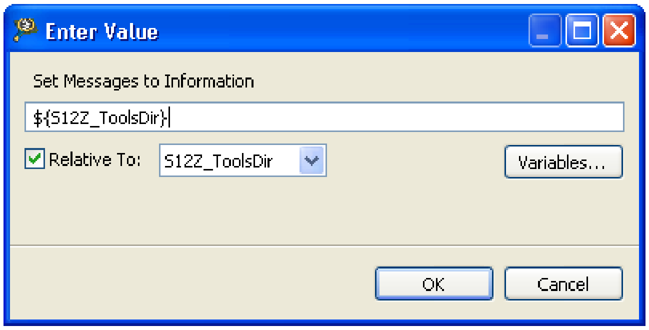
Edit Dialog

Set Messages to Warning

${S12Z_ToolsDir}

Figure 3-47. Set Messages to Warning - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Information** option in the **S12Z Burner > Messages** panel.



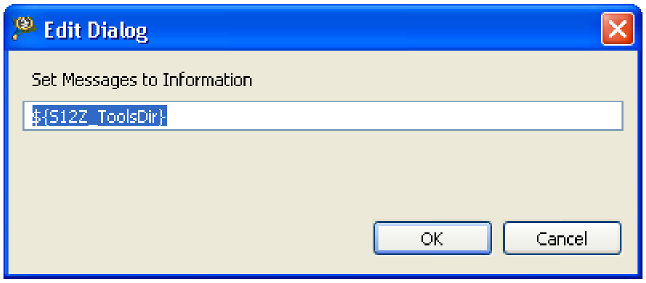
Set Messages to Information

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-48. Set Messages to Information - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Information** option in the **S12Z Burner > Messages** panel.



Set Messages to Information

${S12Z_ToolsDir}

Figure 3-49. Set Messages to Information - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.1.4.1 S12Z Burner > Messages > Disable User Messages

Use this panel to specify the settings for disabling the user messages for the S12Z Burner to use.

The following table lists and describes the disable user messages options for S12Z burner.

Table 3-261. Tool Settings - S12Z Burner > Messages > Disable User Messages Options

Option	Description
Disable all Messages	Use this option to disable all user messages. This option disables messages that are not in the normal message categories like, WARNING, INFORMATION, ERROR, or FATAL by reducing the amount of messages, and simplifying the error parsing of other tools.
Display Type of Messages	Use this option to disable the type of messages.
Disable Informal Messages (e.g. memory model, floating point format)	Use this option to disable the informal messages (e.g., memory model, floating point format).
Disable Processing Statistics Messages (e.g. code size, RAM/ROM usage)	Use this option to disable the messages about processing statistics.
Disable Generated Files Messages	Use this option to disable the messages about the generated files.
Disable Reading Files Messages (e.g. input files)	Use this option to disable the messages about the reading files.
Disable Included Files Messages	Use this option to disable the messages about the include files.

3.11.1.5 S12Z Burner > General

Use this panel to specify other flags for the S12Z Burner to use.

The following table lists and describes the general options for S12Z burner.

Table 3-262. Tool Settings - S12Z Burner > General Options

Option	Description
Other flags	Specify additional command line options for the burner; type in custom flags that are not otherwise available in the UI.

3.11.2 S12Z Linker

Use this panel to specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the general, libraries, and search path settings.

The following table lists and describes the linker options for S12Z.

Table 3-263. Tool Settings - S12Z Linker Options

Option	Description
Command	Shows the location of the linker executable file. Default value is `"${S12Z_ToolsDir}/linker"`. You can specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} ${FLAGS}${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT}`

3.11.2.1 S12Z Linker > Optimization

Use this panel to control linker optimizations. The linker's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the linker optimization options for S12Z **.**

Table 3-264. Tool Settings - S12Z Linker > Optimization Options

Option	Description
Allocation over Segment Boundaries	This option allows you the allocation over segment boundaries. The available options are: • Always Use Next Segment (default) • Always Check for Free Previous Segment • Check for Free Previous Segment when Current is Full
Allocate Non-Referenced Overlap Variables	This option allows you to allocate Non-Referenced Overlap Variables.
Enable Automatic const Placement	With this option the linker constant optimizer is enabled. Instead of performing usual linking actions, the linker generates a data distribution file which contains optimized distribution for constant objects.
Specify Constant Distribution Segment Name	When this option is enabled, it's possible to specify the name of the constant distribution segment.
Allocate Non-Specified const Segments in RAM	This option allocates constant data segments not explicitly allocated in a READ_ONLY segment in the default READ_WRITE segment. This was the default for old versions of the linker, so this option provides a compatible behavior with old linker versions.

Table continues on the next page...

Table 3-264. Tool Settings - S12Z Linker > Optimization Options (continued)

Option	Description
Enable Automatic Data Placement	With this option the linker data optimizer is enabled. Instead of performing usual linking actions, the linker generates a data distribution file which contains optimized distribution.
Specify Data Distribution File Name	When this option is enabled, it's possible to specify the name of the data distribution file. There, all distributed data and how the compiler has to reallocate them are listed.
Generate Data Optimizer Information File	When this option is enabled, the data optimizer generates a data distribution information file giving information on object to segment mapping
Specify Data Distribution Segment Name	When this option is enabled, it's possible to specify the name of the data distribution segment.
Enable Distribution Optimization	This option enables the linker optimizer. Instead of a link, the linker generates a distribution file which contains an optimized distribution.
Specify Distribution File Name	Enable this option to specify the name of the distribution file. The distribution file lists all distributed functions and specifies how the compiler reallocates them.
Generate Optimizer Information File	Using this option, the optimizer generates a distribution information file containing a list of all sections and their functions. Available function information includes the old size, optimized size, and new calling convention.
Choose Optimizing Method	This option allows you to choose the optimizing method. With the FillBanks argument the linker minimizes the free space in every bank. FillBanks is most effective for functions using the near calling convention. Use the CodeSize argument to minimize code when free space within the banks is no concern. The options available are: • Priority is to Fill the Banks (default) • Priority is to Minimize the Code Size
Specify Distribution Segment Name	Use this option to specify the name of the distribution segment.
Specify Library File Name	When this option is enabled,linker generates `file<filename>` which has information about the current libraries and also about the files with which they should be replaced with.
Enable Library Option File Generation	Enables library information generation. When this option is enabled,linker generates file (default `libFile.txt`) which has information about the current library and the startup file and also about the files with which they should be replaced with.
Specify Data Optimizer Options File Name	Specifies the name of the file that contains the set of linker-generated compiler options. When this option is enabled, linker places the second step compiler options in the specified `file<filename>`.
Enable Option File Generation	Enables compiler option generation. The generated options will be used for second step compilation. Linker generates a text file containing a compiler option for the second step (one of the following: `-ConstQualiNear`, -

Table continues on the next page...

Table 3-264. Tool Settings - S12Z Linker > Optimization Options (continued)

Option	Description
	`NonConstQualiNear`, `-Mb`). The content of the file is appended to the compiler options for the second compilation step.
Specify Library File Name	Specifies the name of the library information file. When this option is enabled in second link step,linker reads `file<filename>` which has information about the libraries.
Enable Option to Read libFile.txt in P2	Instructs the linker to read in the library information file that it generated in step one. This option is passed in second link step. It tells the linker to read library information file(default `libFile.txt`).
Emit StartUp Information to Library Info File	The information about the current startup file and the replacement startup file will be added to the library file(default `libFile.txt`) and used during the second compile-link step.
Overlap Constants in ROM	Defines the default if constants and code should be optimized; commands `DO_OVERLAP_CONSTS` and `DO_NOT_OVERLAP_CONSTS` take precedence over the option. The options available are: • No Overlap (default) • Overlap Constant Data and Code • Overlap Constant Data • Overlap Code
Optimize Copy Down	Changes the copy down structure to use few spaces. The optimization does assume that the application does perform both the zero out and the copy down step of the global initialization. If a value is set to zero by the zero out, then zero values are removed from the copy down information. The resulting initialization is not changed by this optimization if the default startup code is used.

3.11.2.2 S12Z Linker > Output

Use this panel to control how the linker formats the listing file, as well as error and warning messages.

The following table lists and describes the linker output options for S12Z.

Table 3-265. Tool Settings - S12Z Linker > Output Options

Option	Description
Link as ROM library	Check to link the application as a ROM library. This option has the same effect as specifying AS ROM_LIB in the linker parameter file.

Table continues on the next page...

Table 3-265. Tool Settings - S12Z Linker > Output Options (continued)

Option	Description
Generate S-Record file	Check to specify that in addition to an absolute file, also an srecord file should be generated. The name of the srecord file is the same as the name of the abs file, except that the extension "SX" is used. The default.env variable "SRECORD" may specify an alternative extension.
Check if Objects Overlap in the Absolute File (even if different address spaces)	Check to instruct the linker to check if objects overlap, taking into account their address space.
Define Default Value of the EPAGE Register	Defines the reset value for the EEPROM Page Index Register (EPAGE). The value is specific to the actual S12(X) derivative.
Define Default Value of the PPAGE Register	Defines the reset value for the Program Page Index Register (PPAGE). The value is specific to the actual S12(X) derivative.
Define Default Value of the RPAGE Register	Defines the reset value for the RAM Page Index Register (RPAGE). The value is specific to the actual S12(X) derivative.
Generate Map File	Check to scan source files for dependencies and emit a Makefile, without generating object code.
Mapping for Memory Space 0x4000-0x7FFF	This option sets the memory mapping for addresses between 0x4000 and 0x7FFF for HCS12XE. This mapping is determined by the MMC control register (the ROMHM and RAMHM bits) and the compiler must be aware of the current setting to correctly perform address translations.
Never Check Section Qualifier Compatibility	For some target CPU's, when placing a section in a segment the linker checks if the qualifiers of the section are compatible with the ones of the segment (for instance when placing .text into RAM may result in a linker error).This option disables the check.
Strip Symbolic Information	Check to disable the generation of DWARF sections in the absolute file to save memory space.
Generate Fixups in abs File	Check to ensure compatibility with previous linker versions. Usually, absolute files do not contain any fixups because all fixups are evaluated at link time. But with fixups, the decoder might symbolically decode the content in absolute files. Some debuggers do not load absolute files which contain fixups because they assume that these fixups are not yet evaluated. But the fixups inserted with this option are actually already handled by this linker.
Enable Stack Consumption Computation	The linker computes maximum stack effect for given application when the option is enabled and places the result in the output .map file.
Specify Statistic File (e.g. statistic.txt)	Specify the name of the linker statistic file. The statistic file reports each allocated object and its attributes. Every attribute is separated by a tab character, so it can be easily imported into a spreadsheet/database program for further processing.

3.11.2.3 S12Z Linker > Input

Use this panel to specify the parameter file path, startup function, object file search paths, and any additional libraries that the C/C++ Linker should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The following table lists and describes the linker input options for S12Z.

Table 3-266. Tool Settings - S12Z Linker > Input Options

Option	Description
Parameter File	Use this option to select the path of the parameter file. Default value is `${ProjDirPath}/Project_Settings/Linker_Files/mc9s12zvmc64.prm.`
Specify Startup Function	Defines the application entry point.
Object File Format	Defines the object file format.
Link Case Insensitive	With this option, the linker ignores object name capitalization. This option supports case-insensitive linking of assembly modules. Since all identifiers are linked case insensitive, this also affects C or C++ modules. This option only affects the comparison of names of linked objects. Section names or the parsing of the link parameter file are unaffected. They remain case sensitive.
Search paths	Shows the list of all search paths; the ELF part of the linker searches object files first in all paths and then the usual environment variables are considered.
Libraries	Lists paths to additional libraries that the C/C++ linker uses. Default value is `"${MCUToolsBaseDir}/S12lisa_Support/s12lisac/lib/ansii.lib"`

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object file search paths.

Table 3-267. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the Add directory path dialog box for the **Search Paths** option in the **S12Z Linker > Input** panel.

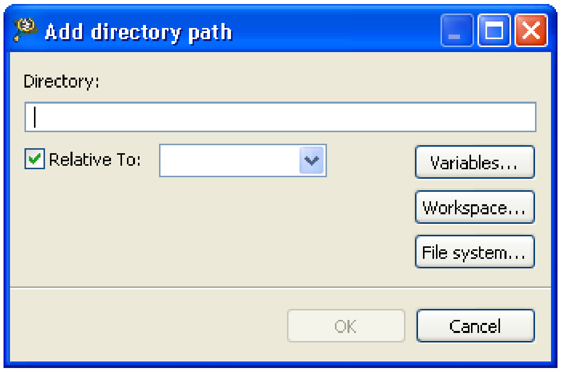
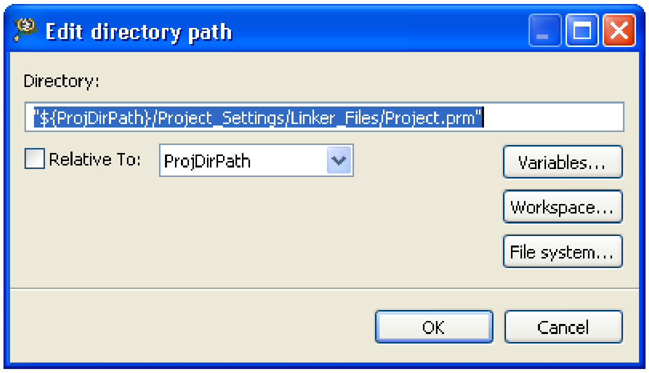


Figure 3-50. Search Paths - Add directory path Dialog Box

The following figure shows the Edit directory path dialog box for the **Search Paths** option in the **S12Z Linker > Input** panel.



Directory: "${ProjDirPath}/Project_Settings/Linker_Files/Project.prm"

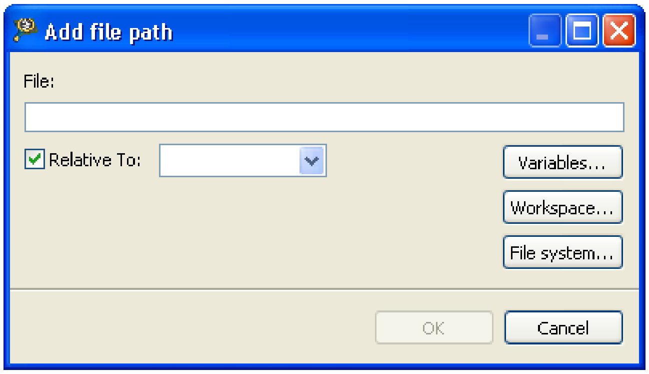
☐ Relative To: ProjDirPath

Figure 3-51. Search Paths - Edit directory path Dialog Box

The buttons in the **Add directory path** and **Edit directory path** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

The following figure shows the Add file path dialog box for the **Libraries** option of the **S12Z Linker > Input** panel.

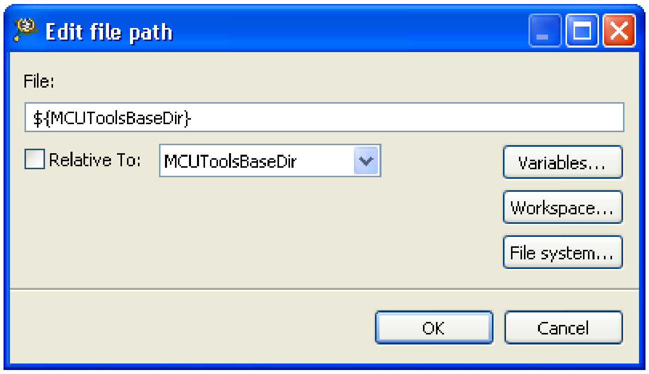


File:

☑ Relative To:

Figure 3-52. Libraries - Add file path Dialog Box

The following figure shows the Edit file path dialog box.



File: ${MCUToolsBaseDir}

□ Relative To: MCUToolsBaseDir

Figure 3-53. Libraries - Edit file path Dialog Box

The buttons in the **Add file path** and **Edit file path** dialog boxes help work with the file paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.2.4 S12Z Linker > Link Order

Use this panel to control the order in which the linker receives the object files.

The following table lists and describes the link order options.

Table 3-268. Tool Settings - Link Order Options

Option	Description
Customize linker input order	Select if you want the linker to receive the object files in the specified order.
Link Order	Lists the object files corresponding to the source files specified in the "link order" list. This option is enables only if Customize linker input order is selected.

3.11.2.5 S12Z Linker > Host

Use this panel to specify the host settings of the S12Z.

The following table lists and describes the memory model options for S12Z.

Table 3-269. Tool Settings - S12Z Linker > Host

Option	Description
Borrow License Feature	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait Until a License is Available from Floating License Server	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	This option allows you to select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the `View` option. • `-ViewWindow`, the application appears with its normal window. • `-ViewMin`, the application appears as an icon in the task bar. • `-ViewMax`, the application appears maximized (filling the whole screen). • `-ViewHidden`, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).

Table continues on the next page...

Table 3-269. Tool Settings - S12Z Linker > Host (continued)

Option	Description
Set Environment Variable	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.

The following table lists and describes the toolbar buttons for the **Set Environment Variable** option.

Table 3-270. Toolbar Buttons - Set Environment Variable Option

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
↑	Move up - Click to move the selected object file search path one position higher in the list.
↓	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Environment Variable** option in the **S12Z Linker > Host** panel.

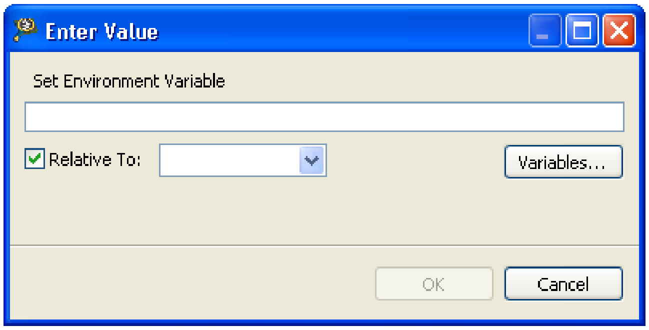


Figure 3-54. Set Environment Variable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Environment Variable** option in the **S12Z Linker > Host** panel.

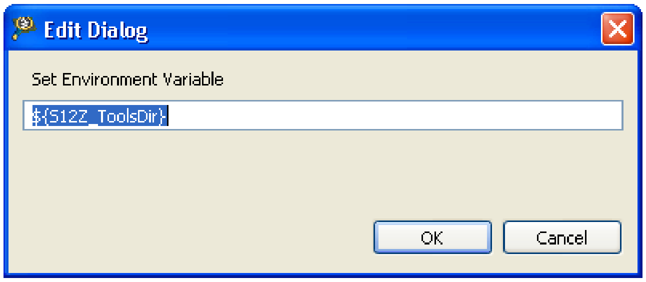


Figure 3-55. Set Environment Variable - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.2.6 S12Z Linker > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-271. Tool Settings - S12Z Linker > Messages Options

Option	Description
Don't Print INFORMATION Messages	This option allows you to disable the INFORMATION messsges. The `-W1` command inhibits the INFORMATION message reporting.
Don't Print INFORMATION or WARNING Messages	This option allows you to disable the printing of INFORMATION or WARNING messages. The `-W2` command suppresses all messages of the type INFORMATION or WARNING.
Create err.log Error File	You can use this option to enable the burner to create the err.log error file. The `-WErrFileOn` command creates or deletes the `err.log` file when the application is finished. When the errors occur, 16-bit window environments use the `err.log` files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the `err.log` file. By default, this checkbox is checked.
Create Error Listing File	You can use this option to create an error listing file. The `-WOutFileOn` command creates an error listing file. This option controls whether an error listing file should be created.

Table continues on the next page...

Table 3-271. Tool Settings - S12Z Linker > Messages Options (continued)

Option	Description
	The error listing file contains a list of all messages and errors that are created during processing. By default, this checkbox is checked.
Cut File Names in Microsoft Format to 8.3	This option truncates the filename to the 8.3 format. The -Wmsg8x3 command cuts the filenames in Microsoft Format to 8.3. Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT.
Set Message File Format for Batch Mode	Use this option to set the message file format for batch mode. The -WmsgFb(-WmsgFbi,-WmsgFbm) command sets the message file format for the batch mode. This option starts the Compiler with additional arguments (for example, files and Compiler options). If you start the Compiler with the arguments (for example, from the Make Tool or with the appropriate argument from an external editor), the Compiler compiles the files in a batch mode. No Compiler window is visible and the Compiler terminates after the job completion. The options available are: • Verbose Format • Microsoft Format (default)
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The -WmsgFob command sets the message format for batch mode. The supported formats are (assuming that the source file is X:\Freescale\mysourcefile.cpph): • %s: Source Extract • %p: Path (example, X:\Freescale\) • %f: Path and name (example, X:\Freescale \mysourcefile) • %n: filename (example, mysourcefile) • %e: Extension (example, .cpph) • %N: File (8 chars) (example, mysource) • %E: Extension (3 chars) (example, .cpp) • %l: Line (example, 3) • %c: Column (example, 47) • %o: Pos (example, 1234) • %K: Uppercase kind (example, ERROR) • %k: Lowercase kind (example, error) • %d: Number (example, C1815) • %m: Message (example, text) • %%: Percent (example, %) • \n: New line • %": A " if the filename, the path, or the extension contains a space • %': A ' if the filename, the path, or the extension contains a space
Message Format for No File Info (e.g. %K %d: %m\n)	Use this option to set the message format for no file information. If there is no file information available for a message, then the <string> in the -WmsgFonf<string> command defines the message format string to use. The supported formats are: • %K: Uppercase kind (example, ERROR) • %k: Lowercase kind (example, error)

Table continues on the next page...

Table 3-271. Tool Settings - S12Z Linker > Messages Options (continued)

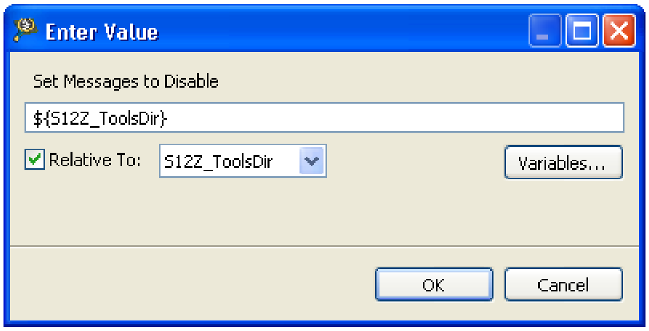
Option	Description
	• %d: Number (example, C1815) • %m: Message (example, text) • %%: Percent (example, %) • \n: New line • %": A " if the filename, if the path or the extension contains a space • %': A ' if the filename, the path or the extension contains a space
Message Format for No Position Info (e.g. %"%f%e%": %K %d: %m\n)	This option allows you to set the message format for no position information. If there is no position information available for a message, then the <string> in the -WmsgFonp<string> command defines the message format string to use. The supported formats are: • %K: Uppercase kind (example, ERROR) • %k: Lowercase kind (example, error) • %d: Number (example, C1815) • %m: Message (example, text) • %%: Percent (example, %) • \n: New line • %": A " if the filename, if the path or the extension contains a space • %': A ' if the filename, the path or the extension contains a space
Maximum Number of Error Messages	This option allows you to set the maximum number of error messages to be displayed. The <number> in the -WmsgNe<number> command sets the number of error messages to be displayed.
Maximum Number of Information Messages	This option allows you to set the amount of information messages that are logged. The <number> in the -WmsgNi<number> command specifies the maximum number of information messages allowed.
Set Messages to Disable	This option allows you to disable the specified messages. The -WmsgSd<number> command sets a message to disable, where <number> is the message number to be disabled, e.g., 1801.
Set Messages to Error	This option changes a message to an error message. The argument <number> in the command -WmsgSe<number> sets the specified message number to be an error, e.g., 1853.
Set Messages to Warning	This option sets a message to a warning message. The argument <number> of the -WmsgSw<number> command, sets the specified error number to be a warning, e.g., 2901.
Set Messages to Information	This option sets a message to an information message. The argument <number> of the command -WmsgSi<number> sets the specified message number to be an information, e.g., 1853.

The following table lists and describes the toolbar buttons for the **Set Messages to Disable** , **Set Messages to Error** , **Set Messages to Warning** and **Set Messages to Information** options of the **S12Z Linker > Messages** panel.

Table 3-272. Toolbar Buttons - Messages Panel

Button	Description
+	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Messages to Disable** option in the **S12Z Linker > Messages** panel.



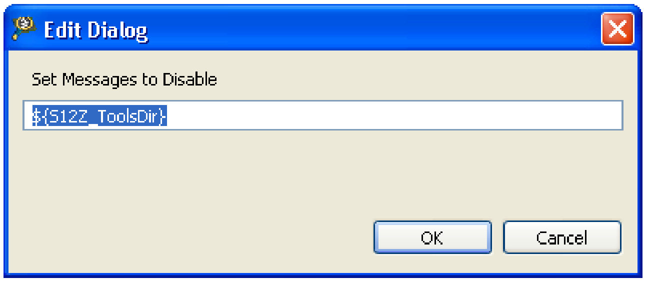
Set Messages to Disable

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-56. Set Messages to Disable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Disable** option in the **S12Z Linker > Messages** panel.

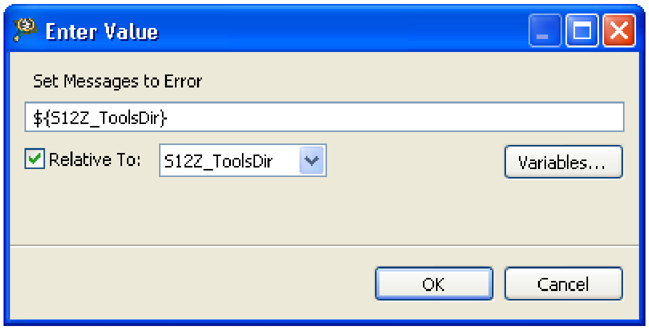


Set Messages to Disable

${S12Z_ToolsDir}

Figure 3-57. Set Messages to Disable - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Error** option in the **S12Z Linker > Messages** panel.



Enter Value

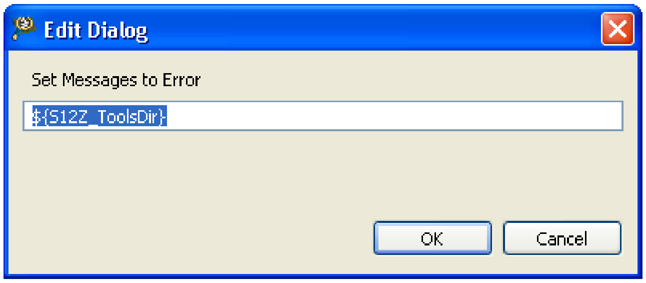
Set Messages to Error

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-58. Set Messages to Error - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Error** option in the **S12Z Linker > Messages** panel.



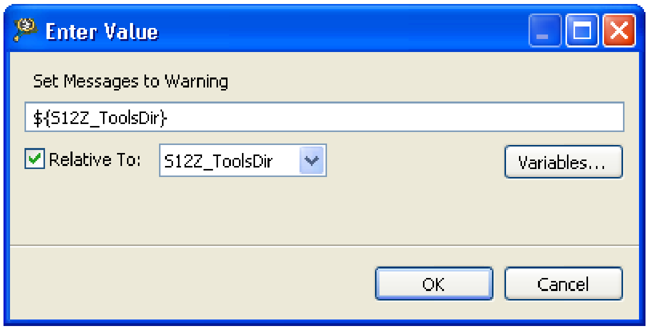
Edit Dialog

Set Messages to Error

${S12Z_ToolsDir}

Figure 3-59. Set Messages to Error - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Warning** option in the **S12Z Linker > Messages** panel.



Enter Value

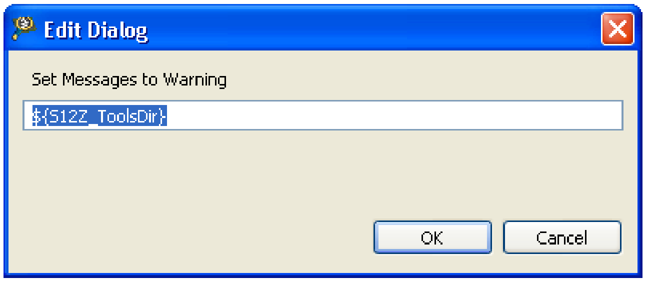
Set Messages to Warning

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-60. Set Messages to Warning - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Warning** option in the **S12Z Linker > Messages** panel.

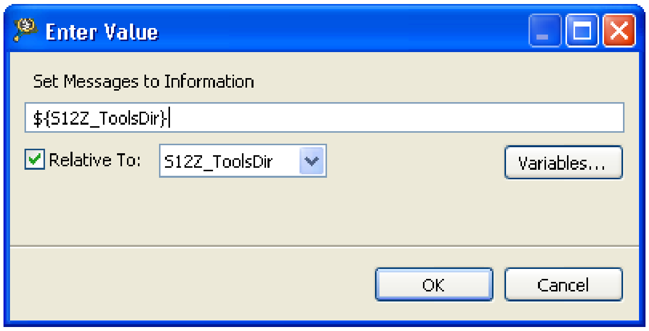


Set Messages to Warning

${S12Z_ToolsDir}

Figure 3-61. Set Messages to Warning - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Information** option in the **S12Z Linker > Messages** panel.



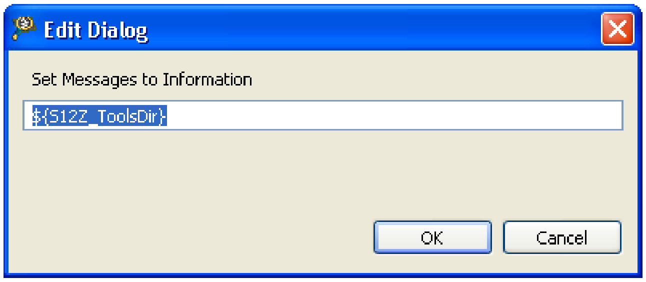
Set Messages to Information

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-62. Set Messages to Information - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Information** option in the **S12Z Linker > Messages** panel.



Set Messages to Information

${S12Z_ToolsDir}

Figure 3-63. Set Messages to Information - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.2.6.1 S12Z Linker > Messages > Disable User Messages

Use this panel to specify the settings for disabling the user messages for the S12Z Linker to use.

The following table lists and describes the disable user messages options for S12Z Linker.

Table 3-273. S12Z Linker > Messages > Disable User Messages Options

Option	Description
Disable all Messages	Use this option to disable all user messages. This option disables messages that are not in the normal message categories like, WARNING, INFORMATION, ERROR, or FATAL by reducing the amount of messages, and simplifying the error parsing of other tools.
Display Type of Messages	Use this option to disable the type of messages.
Disable Informal Messages (e.g. memory model, floating point format)	Use this option to disable the informal messages (e.g., memory model, floating point format).
Disable Included Files Messages	Use this option to disable the messages about the generated files.
Disable Reading Files Messages (e.g. input files)	Use this option to disable the messages about the reading files.
Disable Generated Files Messages	Use this option to disable the messages about the include files.
Disable Processing Statistics Messages (e.g. code size, RAM/ROM usage)	Use this option to disable the messages about processing statistics.

3.11.2.7 S12Z Linker > General

Use this panel to specify the general linker behavior.

The following table lists and describes the general linker options for S12Z.

Table 3-274. Tool Settings - Linker > General Options

Option	Description
Other flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.11.3 S12Z Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the S12Z Compiler tree control includes the general and the file search path settings.

The following table lists and describes the compiler options for S12Z.

Table 3-275. Tool Settings - Compiler Options

Option	Description
Command	Shows the location of the compiler executable file. Default value is: `"${S12Z_ToolsDir}/mwccs12lisa"`. You can specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the compiler will be called with.
Expert Settings	
Command line pattern	Shows the command line pattern; default is `${COMMAND} -c ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.

3.11.3.1 S12Z Compiler > Input

Use this panel to specify file search paths and any additional include files the S12Z Compiler should use. You can specify multiple search paths and the order in which you want to perform the search.

The IDE first looks for an include file in the current directory, or the directory that you specify in the `INCLUDE` directive. If the IDE does not find the file, it continues searching the paths shown in this panel. The IDE keeps searching paths until it finds the `#include` file or finishes searching the last path at the bottom of the Include File Search Paths list. The IDE appends to each path the string that you specify in the `INCLUDE` directive.

NOTE

> The IDE displays an error message if a header file is in a different directory from the referencing source file. Sometimes, the IDE also displays an error message if a header file is in the same directory as the referencing source file.

For example, if you see the message `Could not open source file myfile.h`, you must add the path for `myfile.h` to this panel.

The following table lists and describes the input options for S12Z Compiler.

Table 3-276. Tool Settings - S12Z Compiler > Input Options

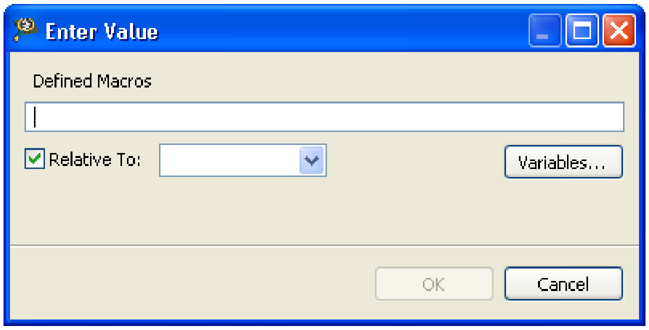
Option	Description
Prefix File	This option allows you to specify the prefix file or precompiled header file search path.
Source File Encoding	This option allows you to select the source file encoding. The options available are: • ASCII (default) • Auto-Detect (multibyte encoding) • System (use system locale) • UTF-8 • Shift-JIS • EUC-JP • ISO-2022-JP
Allow Macro Redefinition	This option allows macro redefinitions without an error or warning.
Defined Macros	Use this option to specify the defined macros.
Undefined Macros	Use this option to specify the undefined macros.

The following table lists and describes the toolbar buttons that help work with the file paths.

Table 3-277. Defined Macros/Undefined Macros Toolbar Buttons

Button	Description
➕	Add - Click to open the Enter Value dialog box and specify location of the library you want to add.
✖	Delete - Click to delete the selected library path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected path.
⇧	Move up - Click to move the selected path one position higher in the list.
⇩	Move down - Click to move the selected path one position lower in the list.

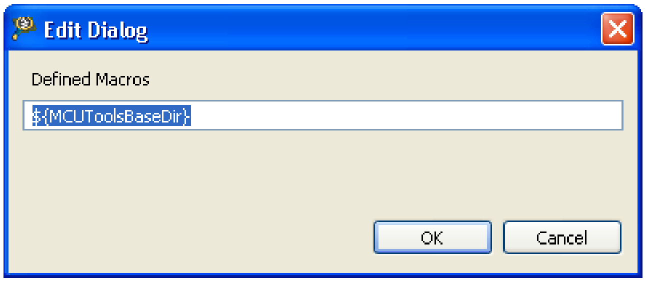
The following figure shows the **Enter Value** dialog box for **Defined Macros** .



☑ Relative To:

Figure 3-64. Tool Settings - S12Z Compiler > Defined Macros > Enter value Dialog Box

The following figure shows the **Edit Dialog** dialog box for **Defined Macros** .



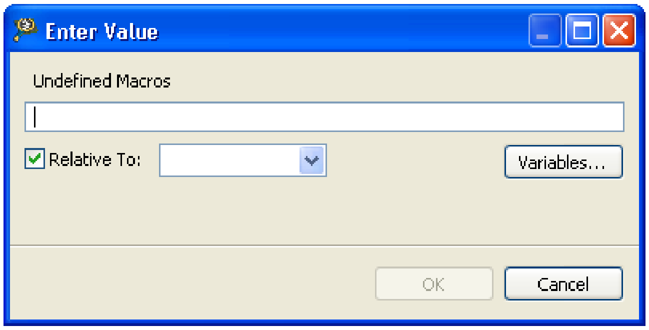
${MCUToolsBaseDir}

Figure 3-65. Defined Macros - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

The following figure shows the **Enter Value** dialog box for **Undefined Macros** .



☑ Relative To:

Figure 3-66. Undefined Macros - Enter value Dialog Box

The following figure shows the **Edit Dialog** dialog box for **Undefined Macros** .

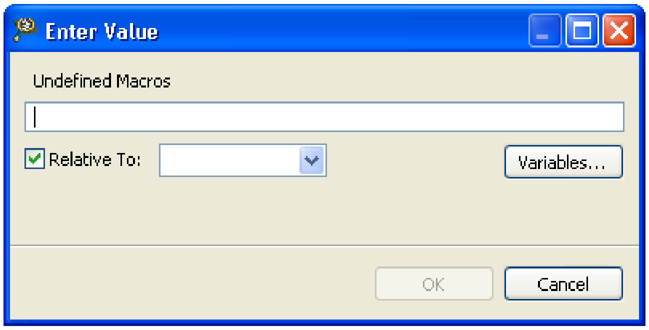


Figure 3-67. Undefined Macros - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.3.2 S12Z Compiler > Access Paths

Use this panel to specify access path options for the S12Z Compiler.

The following table lists and describes the access paths options for S12Z.

Table 3-278. Tool Settings - HCS08 Compiler > Language Options

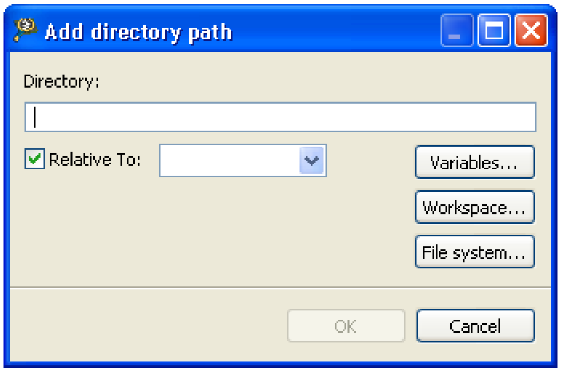
Option	Description
Do Not use MWCIncludes Variable	This option inhibits the usage of MWCInclude variables. By default, this checkbox is checked.
Always Search User Paths	Use this option to enable the usage of the search paths.
Search User Paths (#include "...")	Use this option to specify the user paths.
Search User Paths Recursively	Use this option to specify the user paths recursively.
Search System Paths (#include <...>)	Use this option to specify the system paths.
Search System Paths Recursively	Use this option to specify the system paths recursively.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object file search paths.

Table 3-279. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

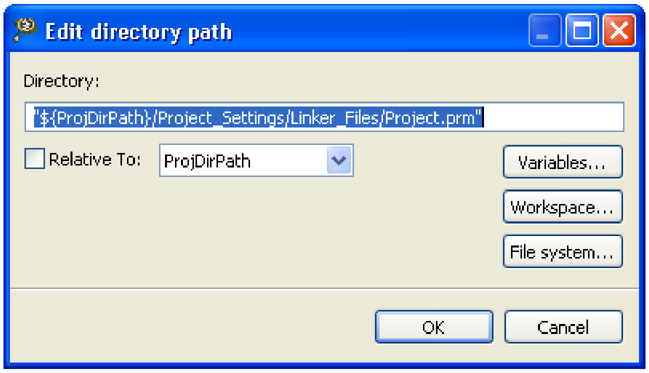
The following figure shows the Add directory path dialog box.



☑ Relative To:

Figure 3-68. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



Directory: "${ProjDirPath}/Project_Settings/Linker_Files/Project.prm"

☐ Relative To: ProjDirPath

Figure 3-69. Edit directory path Dialog Box

The buttons in the **Add directory path** and **Edit directory path** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **Folder selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.3.3 S12Z Compiler > Warnings

Use this panel to specify the warnings settings for S12Z compiler.

The following table lists and describes the **Warnings** options for S12Z compiler.

Table 3-280. Tool Settings - S12Z Compiler > Warnings Options

Option	Description
Treat All Warnings as Errors	Check to treat all warnings as errors. The compiler will stop if it generates a warning message.
Enable Warnings	Select the level of warnings you want reported from the compiler. Custom lets you to select individual warnings. Other settings select a pre-defined set of warnings.
Illegal #pragmas (most)	Check to notify the presence of illegal pragmas.
Possible Unwanted Effects (most)	Check to notify the presence of illegal pragmas.
Extended Error Checks (most)	Check if you want to do an extended error checking.
Hidden Virtual Functions (most)	Check to generate a warning message if you declare a non-virtual member function that prevents a virtual function, that was defined in a superclass, from being called and is equivalent to pragma `warn_hidevirtual` and the command-line option `-warnings hidevirtual`.
Implicit Arithmetic Conversions (all)	Check to warn of implict arithmetic conversions.
Implicit Signed/Unsigned Conversions (all)	Check to enable warning of implict conversions between signed and unsigned variables.
Implicit Float to Integer Conversions (all)	Check to warn of implict conversions of a floating-point variable to integer type.
Implicit Integer to Float Conversions (all)	Check to warn of implict conversion of an integer variable to floating-point type.
Pointer/Integer Conversions (most)	Check to enable warnings of conversions between pointer and integers.
Unused Arguments (most)	Check to warn of unused arguments in a function.
Unused Variables (most)	Check to warn of unused variables in the code.

Table continues on the next page...

Table 3-280. Tool Settings - S12Z Compiler > Warnings Options (continued)

Option	Description
Unused Result from Non-Void-Returning Function (full)	Check to warn of unused result from nonvoid-returning functions.
Missing 'return' Value in Non-Void-Returning Function (most)	Check to warn of when a function lacks a return statement.
Expression Has No Side Effect (most)	Check to issue a warning message if a source statement does not change the program's state. This is equivalent to the pragma `warn_no_side_effect`, and the command-line option `-warnings unusedexpr`.
Extra Commas (most)	Check to issue a warning message if a list in an enumeration terminates with a comma. The compiler ignores terminating commas in enumerations when compiling source code that conforms to the ISO/IEC 9899-1999 ("C99") standard and is equivalent to pragma `warn_extracomma` and the command-line option `-warnings extracomma`.
Empty Declarations (most)	Check to warn of empty declarations.
Inconsistent 'class'/'struct' Usage (most)	Check to warn of inconsistent usage of class or struct.
Incorrect Capitalization in #include "..." (most)	Check to issue a warning message if the name of the file specified in a `#include "file"` directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps` and the commandline option `-warnings filecaps`.
Incorrect Capitalization in System #include <...> (most)	Check to issue a warning message if the name of the file specified in a `#include <file>` directive uses different letter case from a file on disk and is equivalent to pragma `warn_filenamecaps_system` and the command-line option - `warnings sysfilecaps`.
Pad Bytes Added (full)	Check to issue a warning message when the compiler adjusts the alignment of components in a data structure and is equivalent to pragma `warn_padding` and the command-line option `-warnings padding`.
Undefined Macro in #if/#elif (full)	Check to issues a warning message if an undefined macro appears in `#if` and `#elif` directives and is equivalent to pragma `warn_undefmacro` and the command-line option `-warnings undefmacro`.
Non-Inlined Functions (full)	Check to issue a warning message if a call to a function defined with the `inline`, `__inline__`, `or` `__inline` keywords could not be replaced with the function body and is equivalent to pragma `warn_notinlined` and the command-line option `-warnings notinlined`.
Token Not Formed by ## Operator (most)	Check to enable warnings for the illegal uses of the preprocessor's token concatenation operator (##). It is equivalent to the pragma `warn_illtokenpasting on`.

3.11.3.4 S12Z Compiler > Code Generation

Use this panel to specify the code generation compiler behavior.

The following table lists and describes the code generation compiler options for S12Z.

Table 3-281. Tool Settings - S12Z Compiler > Code Generation Options

Option	Description
Memory Model	This option allows to specify the memory model. The options available are: Small, Medium (default), and Large.
Bit field byte allocation from LSB to MSB (right-to-left)	By default, bit allocation in byte bitfields proceeds from the least significant bit to the most significant bit. This produces less code overhead in the case of partially- allocated byte bitfields. • MSB: Most significant bit in byte first (left to right) • LSB: Least significant bit in byte first (right to left)
Bit field type size reduction	This option is configurable whether or not the compiler uses type-size reduction for bitfields. Type-size reduction means that the compiler can reduce the type of an int bitfield to a char bitfield if it fits into a character. This allows the compiler to allocate memory only for one byte instead of for an integer.

3.11.3.5 S12Z Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The following table lists and describes the **Optimization** options for S12Z compiler.

Table 3-282. Tool Settings - S12Z Compiler > Optimization Options

Option	Description
Optimization Level	This option is disabled for S12Z Compiler.
Speed Vs Size	This option allows you to specify the type of optimization. The options available are: • Speed • Size
Memory Model	This option allows to specify the memory model. The options available are: • Small • Medium (default) • Large
Inline Level	Enables inline expansion. If there is a #pragma INLINE before a function definition, all calls of this function are replaced by the code of this function, if possible. The options available are: • Off • Smart (default) • 1

Table continues on the next page...

Table 3-282. Tool Settings - S12Z Compiler > Optimization Options (continued)

Option	Description
	• 2 • 3 • 4 • 5 • 6 • 7 • 8
Auto Inline	Check to enable the auto inlining.
Bottom-Up Inlining	Check to control the bottom-up function inlining method. When active, the compiler inlines function code starting with the last function in the chain of functions calls, to the first one.

3.11.3.6 S12Z Compiler > Language

Use this panel to specify the language settings for S12Z compiler.

The following table lists and describes the **Language** options for S12Z compiler.

Table 3-283. Tool Settings - S12Z Compiler > Language Options

Option	Description
Require Function Prototypes	Check to enforce the requirement of function prototypes. The compiler generates an error message if you define a previously referenced function that does not have a prototype. If you define the function before it is referenced but do not give it a prototype, this setting causes the compiler to issue a warning message.
Enable C++ 'bool' type, 'true' and 'false' Constants	Check to enable the C++ compiler to recognize the bool type and its true and false values specified in the ISO/IEC 14882-1998 C++ standard.
ISO C++ Template Parser	Check to follow the ISO/IEC 14882-1998 standard for C++ to translate templates, enforcing more careful use of the typename and template keywords. The compiler also follows stricter rules for resolving names during declaration and instantiation.
Use Instance Manager	Check to reduce compile time by generating any instance of a C++ template (or noninlined inline) function only once.
Force C++ Compilation	Check to enable the forced C++ compilation.
Enable GCC Extensions	Check to recognize language features of the GNU Compiler Collection (GCC) C compiler that are supported by CodeWarrior compilers; is equivalent to pragma `gcc_extensions` and the command-line option `-gcc_extensions`.

Table continues on the next page...

Table 3-283. Tool Settings - S12Z Compiler > Language Options (continued)

Option	Description
Enable C99 Extensions	Check to recognize ISO/IEC 9899-1999 ("C99") language features; is equivalent to pragma `c99` and the command-line option `-dialect c99`.
Enable C++ Exceptions	Check to generate executable code for C++ exceptions; is equivalent to pragma `exceptions` and the command-line option `-cpp_exceptions`.
Enable RTTI	Check to allow the use of the C++ runtime type information (RTTI) capabilities, including the `dynamic_cast` and typeid operators; is equivalent to `pragma RTTI` and the command-line option `-RTTI`.
Enable wchar_t Support	Check to enable C++ compiler recognize the `wchar_t` data type specified in the ISO/IEC 14882-1998 C++ standard; is equivalent to `pragma wchar_type` and the command-line option `-wchar_t`.
ANSI Strict	Check to enable C compiler operate in strict ANSI mode. In this mode, the compiler strictly applies the rules of the ANSI/ ISO specification to all input files. This setting is equivalent to specifying the - ansi command-line option. The compiler issues a warning for each ANSI/ISO extension it finds.
ANSI Keywords Only	Check to generate an error message for all non-standard keywords (ISO/IEC 9899-1990 C, ï¿½6.4.1). If you must write source code that strictly adheres to the ISO standard, enable this setting; is equivalent to pragma `only_std_keywords` and the commandlineoption `-stdkeywords`.
Expand Trigraphs	Check to recognize trigraph sequences (ISO/ IEC 9899-1990 C, ï¿½5.2.1.1); is equivalent to pragma `trigraphs` and the commandline option `-trigraphs`.
Legacy for-scoping	Check to generate an error message when the compiler encounters a variable scope usage that the ISO/IEC 14882-1998 C++ standard disallows, but is allowed in the C++ language specified in The Annotated C++ Reference Manual ("ARM"); is equivalent to pragma ARM_scoping and the commandline option `-for_scoping`.
Enum Always Int	Check to use signed integers to represent enumerated constants and is equivalent to pragma `enumsalwaysint` and the command-line option `-enum`.
Use Unsigned Chars	Check to treat char declarations as unsigned char declarations and is equivalent to pragma `unsigned_char` and the command-line option `-char unsigned`.
Reuse Strings	Check to store only one copy of identical string literals and is equivalent to opposite of the pragma `dont_reuse_strings` and the command-line option `-string reuse`.
Pool Strings	Check to collect all string constants into a single data section in the object code it generates and is equivalent to pragma `pool_strings` and the command-line option `-strings pool`.

3.11.3.7 S12Z Compiler > Messages

Use this panel to specify the messages settings for S12Z compiler.

The following table lists and describes the **Messages** options for S12Z compiler.

Table 3-284. Tool Settings - S12Z Compiler > Messages Options

Option	Description
Message Style	Use this option to set the message style. The options available are: • GCC • MPW • Standard • IDE • Parseable (default) • Enterprise-IDE
Maximum Number of Errors	This option allows you to specify the maximum number of error messages to be displayed.
Maximum Number of Warnings	This option allows you to specify the maximum number of warning messages to be displayed.

3.11.3.8 S12Z Compiler > General

Use this panel to specify the general compiler behavior.

The following table lists and describes the general compiler options for S12Z.

Table 3-285. Tool Settings - Linker > General Options

Option	Description
Generate Debug Information	This option allows the compiler to generate the debug information. By default, this checkbox is checked.
Other Flags	Specify additional command line options for the linker; type in custom flags that are not otherwise available in the UI.

3.11.4 S12Z Assembler

Use this panel to specify the command, options, and expert settings for the S12Z build tool assembler.

The following table lists and describes the assembler options for S12Z.

Table 3-286. Tool Settings - S12Z Assembler Options

Option	Description
Command	Shows the location of the assembler executable file. You can specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`.
Command line pattern	

3.11.4.1 S12Z Assembler > Output

Use this panel to specify file search paths and any additional include files the S12Z Assembler should use. You can specify multiple search paths and the order in which you want to perform the search.

The following table lists and describes the input options for S12Z Assembler.

Table 3-287. Tool Settings - S12Z Assembler > Output Options

Option	Description
Object File Format	The Assembler writes the code and debugging info after compilation into an object file. The Assembler uses a HIWARE-proprietary object-file format when the -Fh, -F6, or -F7 options are set. The Assembler produces an ELF/DWARF object file when the -F1 or -F2 options are set. This object-file format may also be supported by other Assembler vendors. In the Assembler ELF/DWARF 2.0 output, some constructs written in previous versions were not conforming to the ELF standard because the standard was not clear enough in this area. Because old versions of the simulator or debugger (V5.2 or earlier) are not able to load the corrected new format, the old behavior can still be produced by using -f2o instead of -f2. Some old versions of the debugger (simulator or debugger V5.2 or earlier) generate a GPF when a new absolute file is loaded. If you want to use the older versions, use -f2o instead of -f2. New versions of the debugger are able to load both formats correctly. Also, some older ELF/DWARF object file loaders from emulator vendors may require you to set the -F2o option. The -F1o option is only supported if the target supports the ELF/DWARF 1.1 format. This option is only used with older debugger versions as a compatibility option. The available options are: • ELF/DWARF 2.0 Object File Format • ELF/DWARF 2.0 Absolute File Format

Table continues on the next page...

Table 3-287. Tool Settings - S12Z Assembler > Output Options (continued)

Option	Description
	- Compatible ELF/DWARF 2.0 Object File Format - Compatible ELF/DWARF 2.0 Absolute File Format - HIWARE Object File Format
Show Label Statistics	The `-Ll` option causes the Compiler to append statistical information about the compilation session to the specified file. Compiler options, code size (in bytes), stack usage (in bytes) and compilation time (in seconds) are given for each procedure of the compiled file. The information is appended to the specified filename (or the file 'make.txt', if no argument given).
Generate Listing File (e.g. %(TEXTPATH)/%n.lst)	This option enables the Compiler to generate the listing files.
Address Size in the Listing File (integer)	This option allows you to specify the address size in the listing file.
Do Not Print Macro Call in Listing File	This option disables the printing of the macro call in listing the file.
Do Not Print Macro Definition in Listing File	This option disables the printing of the macro definition in the listing file.
Do Not Print Macro Expansion in Listing File	This option disables the printing of the macro expansion in the listing file.
Do Not Print Included Files in Listing File	This option disables the printing of the included files in the listing file.

3.11.4.1.1 S12Z Assembler > Output > Configure Listing File

Use this panel to specify the S12Z assembler's listing file configuration settings.

The following table lists and describes the configure listing file options for S12Z.

Table 3-288. Tool Settings - S12Z Assembler > Output > Configure Listing File

Option	Description
Select All	This option prints all the columns in the listing file.
Do Not Write the Source Line	This option disables the printing of the source column in the listing file.
Do Not Write the Relative Line	This option disables the printing of the relative column in the listing file.
Do Not Write the Macro Mark	This option disables the printing of the macro mark column in the listing file.
Do Not Write the Address	This option disables the printing of the address coulumn in the listing file.
Do Not Write the Location Kind	This option disables the printing of the the location type column in the listing file.
Do Not Write the Include Mark Column	This option disables the printing of the include mark column in the listing file.

Table continues on the next page...

Table 3-288. Tool Settings - S12Z Assembler > Output > Configure Listing File (continued)

Option	Description
Do Not Write the Object Code	This option disables the printing of the object code in the listing file.
Do Not Write the Absolute Line	This option disables the printing of the absolute lines in the listing file.

3.11.4.2 S12Z Assembler > Input

Use this panel to specify the input settings for S12Z assembler.

The following table lists and describes the input options for S12Z.

Table 3-289. Tool Settings - S12Z Assembler > Input

Option	Description
Case Insensitivity on Label Name	Use this option to enable the case insensitivity on the label name.
Define Label (use spaces to separate labels)	Use this option to specify the label names.
Support for Structured Types	This option enables the support for the structured types.
Include File Search Path	Use this option to specify the include file search path.

The following table lists and describes the toolbar buttons that help work with the libraries and the additional object file search paths.

Table 3-290. Search Paths Toolbar Buttons

Button	Description
➕	Add - Click to open the **Add directory path** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit directory path** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the Add directory path dialog box.

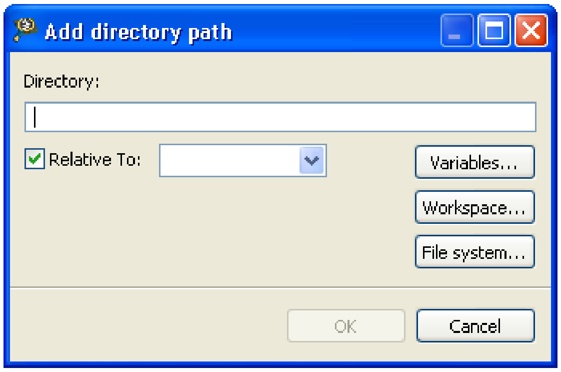
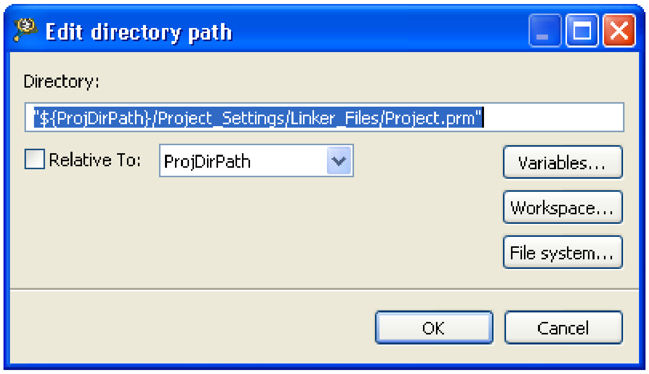


Figure 3-70. Add directory path Dialog Box

The following figure shows the Edit directory path dialog box.



"${ProjDirPath}/Project_Settings/Linker_Files/Project.prm"

ProjDirPath

Figure 3-71. Edit directory path Dialog Box

The buttons in the **Add directory path** and **Edit directory path** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- Workspace **-** Click to display the **Folder Selection** dialog box and specify the variable for object file search path. The resulting variable, relative to the workspace, appears in the appropriate list.
- File system **-** Click to display the **Browse for Folder** dialog box and specify the object file search path. The resulting path appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.4.3 S12Z Assembler > Language

Use this panel to specify language options for the S12Z Assembler.

The following table lists and describes the language options for S12Z Assembler.

Table 3-291. Tool Settings - S12Z Assembler > Language Options

Option	Description
Angle Brackets for Macro Arguments Grouping	Controls whether the < > syntax for macro invocation argument grouping is available. When it is disabled, the Assembler does not recognize the special meaning for < in the macro invocation context.
Square Braces for Macro Arguments Grouping	Controls the availability of the [? ?] syntax for macro invocation argument grouping. When it is disabled, the Assembler does not recognize the special meaning for [? in the macro invocation context.
Maximum Macro Nesting	Controls how deep macros calls can be nested. Its main purpose is to avoid endless recursive macro invocations.

3.11.4.3.1 S12Z Assembler > Language > Compatibility modes

Use this panel to specify language compatibility modes for the HCS08 Assembler.

The following table lists and describes the compatibility mode options for HCS08 Assembler.

Table 3-292. Tool Settings - S12Z Assembler > Language > Compatibility mode Options

Option	Description
Select All	Check to enable all compatibility mode options.
Symbol Prefixes	With this suboption, the Assembler accepts "pgz:" and "byte:" prefixed for symbols in XDEFs and XREFs. They correspond to XREF.B or XDEF.B with the same symbols without the prefix.
Ignore FF Character at Line Start	With this suboption, an otherwise improper character recognized from feed character is ignored.
Alternate Comment Rules	With this suboption, comments implicitly start when a space is present after the argument list. A special character is not necessary. Be careful with spaces when this option is given because part of the intended arguments may be taken as a comment. However, to avoid accidental comments, the Assembler does issue a warning if such a comment does not start with a "*" or a ";".
Support FOR Directive	With this suboption, the Assembler supports a FOR - Repeat assembly block assembly directive to generate repeated patterns more easily without having to use recursive macros.
Support Additional Directives	With this suboption, some additional directives are added for enhanced compatibility.

Table continues on the next page...

Table 3-292. Tool Settings - S12Z Assembler > Language > Compatibility mode Options (continued)

Option	Description
Operator != Means Equal	The Assembler takes the default value of the != operator as not equal, as it is in the C language. For compatibility, this behavior can be changed to equal with this option. Because of the risks involved with this option for existing code, a message is issued for every != which is treated as equal.
Support $ Character in Symbols	With this suboption, the Assembler supports to start identifiers with a $ sign.
Support Additional ! Operators	With this suboption, the Assembler supports to start the identifiers with a ! sign.

3.11.4.4 S12Z Assembler > Host

Use this panel to specify the host settings of the S12Z assembler.

The following table lists and describes the memory model options for S12Z.

Table 3-293. Tool Settings - S12Z Assembler > Host

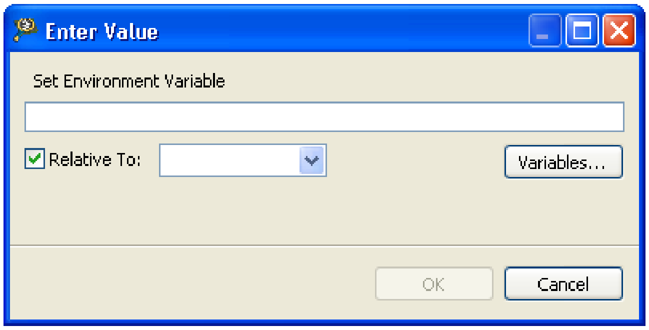
Option	Description
Borrow License Feature	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait Until a License is Available from Floating License Server	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	This option allows you to select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the View option. • `-ViewWindow`, the application appears with its normal window. • `-ViewMin`, the application appears as an icon in the task bar. • `-ViewMax`, the application appears maximized (filling the whole screen). • `-ViewHidden`, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).
Set Environment Variable	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.

The following figure lists and describes the toolbar buttons for the **Set Environment Variable** option.

Table 3-294. Toolbar Buttons - Set Environment Variable Option

Button	Description
+	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Environment Variable** option in the **S12Z Assembler > Host** panel.

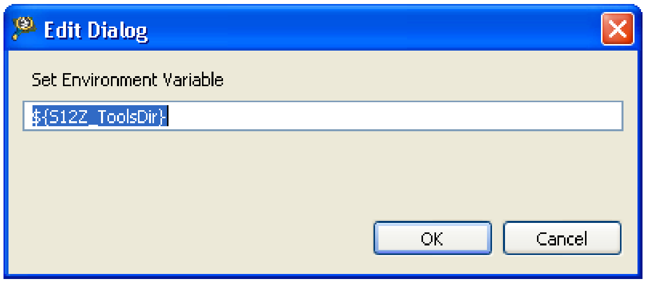


Set Environment Variable

☑ Relative To:

Figure 3-72. Set Environment Variable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Environment Variable** option in the **S12Z Assembler > Host** panel.



Set Environment Variable

${S12Z_ToolsDir}

Figure 3-73. Set Environment Variable - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.4.5 S12Z Assembler > Code Generation

Use this panel to specify the code generation assembler behavior.

The following table lists and describes the code generation assembler options for S12Z.

Table 3-295. Tool Settings - S12Z Assembler > Code Generation Options

Option	Description
Associate Debug Information to Assembly Source File	Passes the assembly source file name information to DWARF sections. When the output .abs file is debugged, the actual assembly source file is displayed instead of intermediary <filename>.dbg file.

3.11.4.6 S12Z Assembler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-296. Tool Settings - S12Z Assembler > Messages Options

Option	Description
Don't Print INFORMATION Messages	This option allows you to disable the INFORMATION messsges. The `-W1` command inhibits the INFORMATION message reporting.
Don't Print INFORMATION or WARNING Messages	This option allows you to disable the printing of INFORMATION or WARNING messages. The `-W2` command suppresses all messages of the type INFORMATION or WARNING.
Create err.log Error File	You can use this option to enable the burner to create the err.log error file. The `-WErrFileOn` command creates or deletes the `err.log` file when the application is finished. When the errors occur, 16-bit window environments use the `err.log` files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the `err.log` file. By default, this checkbox is checked.

Table continues on the next page...

Table 3-296. Tool Settings - S12Z Assembler > Messages Options (continued)

Option	Description
Create Error Listing File	You can use this option to create an error listing file. The `-WOutFileOn` command creates an error listing file. This option controls whether an error listing file should be created. The error listing file contains a list of all messages and errors that are created during processing. By default, this checkbox is checked.
Cut File Names in Microsoft Format to 8.3	This option truncates the filename to the 8.3 format. The `-Wmsg8x3` command cuts the filenames in Microsoft Format to 8.3. Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT.
Set Message File Format for Batch Mode	Use this option to set the message file format for batch mode. The `-WmsgFb(-WmsgFbi,-WmsgFbm)` command sets the message file format for the batch mode. This option starts the Compiler with additional arguments (for example, files and Compiler options). If you start the Compiler with the arguments (for example, from the Make Tool or with the appropriate argument from an external editor), the Compiler compiles the files in a batch mode. No Compiler window is visible and the Compiler terminates after the job completion. The options available are: • Verbose Format • Microsoft Format (default)
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`) • `%E`: Extension (3 chars) (example, `.cpp`) • `%l`: Line (example, `3`) • `%c`: Column (example, `47`) • `%o`: Pos (example, `1234`) • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, the path, or the extension contains a space • `%'`: A ' if the filename, the path, or the extension contains a space

Table continues on the next page...

Table 3-296. Tool Settings - S12Z Assembler > Messages Options (continued)

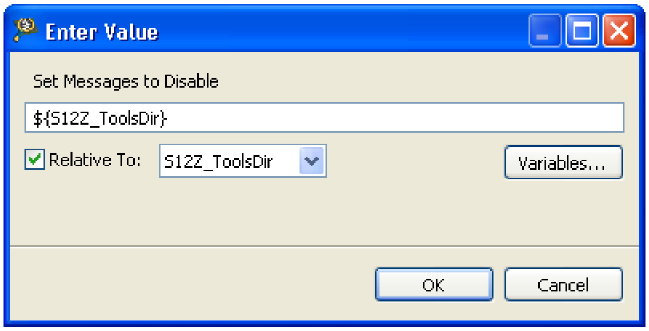
Option	Description
Message Format for No File Info (e.g. %K %d: %m\n)	Use this option to set the message format for no file information. If there is no file information available for a message, then the `<string>` in the `-WmsgFonf<string>` command defines the message format string to use. The supported formats are: • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, if the path or the extension contains a space • `%'`: A ' if the filename, the path or the extension contains a space
Message Format for No Position Info (e.g. %"%f%e%": %K %d: %m\n)	This option allows you to set the message format for no position information. If there is no position information available for a message, then the `<string>` in the `-WmsgFonp<string>` command defines the message format string to use. The supported formats are: • `%K`: Uppercase kind (example, ERROR) • `%k`: Lowercase kind (example, error) • `%d`: Number (example, C1815) • `%m`: Message (example, text) • `%%`: Percent (example, %) • `\n`: New line • `%"`: A " if the filename, if the path or the extension contains a space • `%'`: A ' if the filename, the path or the extension contains a space
Maximum Number of Error Messages	This option allows you to set the maximum number of error messages to be displayed. The `<number>` in the `-WmsgNe<number>` command sets the number of error messages to be displayed.
Maximum Number of Information Messages	This option allows you to set the amount of information messages that are logged. The `<number>` in the `-WmsgNi<number>` command specifies the maximum number of information messages allowed.
Set Messages to Disable	This option allows you to disable the specified messages. The `-WmsgSd<number>` command sets a message to disable, where `<number>` is the message number to be disabled, e.g., 1801.
Set Messages to Error	This option changes a message to an error message. The argument `<number>` in the command `-WmsgSe<number>` sets the specified message number to be an error, e.g., 1853.
Set Messages to Warning	This option sets a message to a warning message. The argument `<number>` of the `-WmsgSw<number>` command, sets the specified error number to be a warning, e.g., 2901.
Set Messages to Information	This option sets a message to an information message. The argument `<number>` of the command `-WmsgSi<number>` sets the specified message number to be an information, e.g., 1853.

The following table lists and describes the toolbar buttons for the **Set Messages to Disable** , **Set Messages to Error** , **Set Messages to Warning** and **Set Messages to Information** options of the **S12Z Assembler > Messages** panel.

Table 3-297. Search Paths Toolbar Buttons - Messages Panel

Button	Description
	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
	Delete - Click to delete the selected object file search path.
	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
	Move up - Click to move the selected object file search path one position higher in the list.
	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Messages to Disable** option in the **S12Z Assembler > Messages** panel.



Enter Value

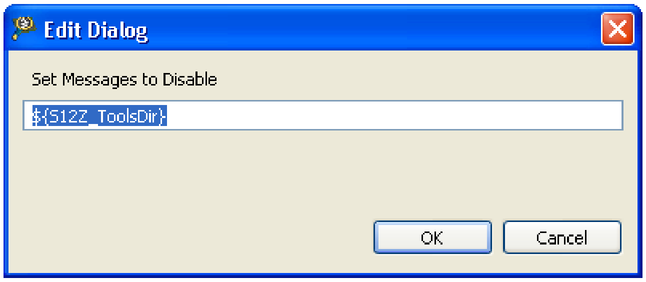
Set Messages to Disable

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-74. Set Messages to Disable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Disable** option in the **S12Z Assembler > Messages** panel.



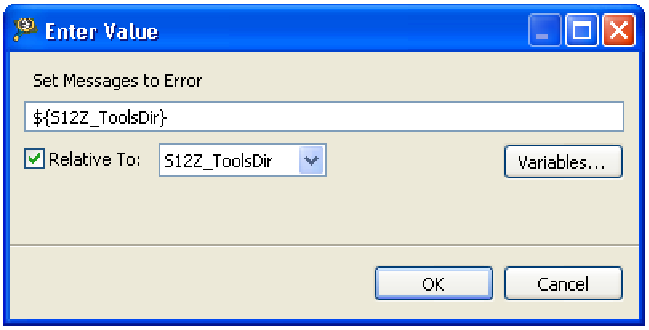
Edit Dialog

Set Messages to Disable

${S12Z_ToolsDir}

Figure 3-75. Set Messages to Disable - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Error** option in the **S12Z Assembler > Messages** panel.



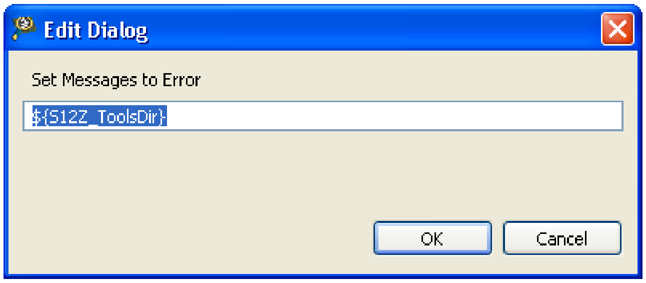
Set Messages to Error

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-76. Set Messages to Error - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Error** option in the **S12Z Assembler > Messages** panel.

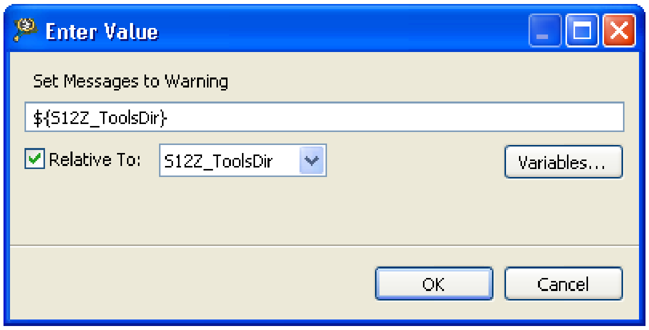


Set Messages to Error

${S12Z_ToolsDir}

Figure 3-77. Set Messages to Error - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Warning** option in the **S12Z Assembler > Messages** panel.



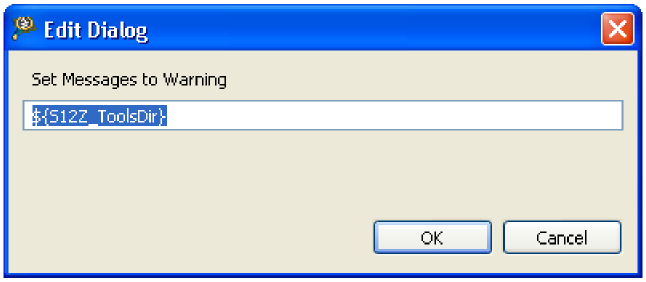
Set Messages to Warning

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-78. Set Messages to Warning - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Warning** option in the **S12Z Assembler > Messages** panel.

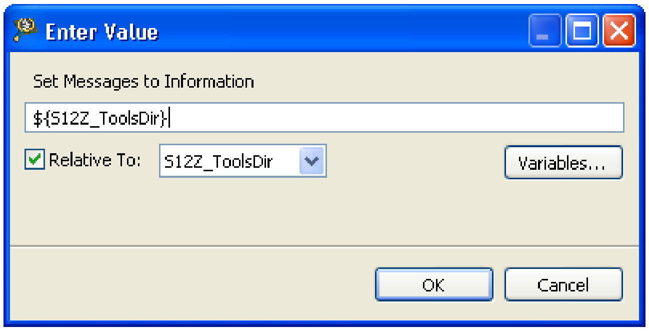


Set Messages to Warning

${S12Z_ToolsDir}

Figure 3-79. Set Messages to Warning - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Information** option in the **S12Z Assembler > Messages** panel.



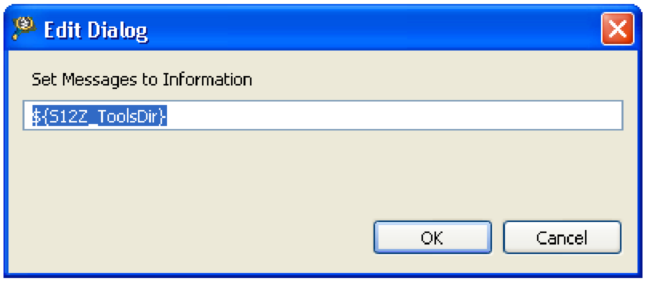
Set Messages to Information

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-80. Set Messages to Information - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Information** option in the **S12Z Assembler > Messages** panel.



Set Messages to Information

${S12Z_ToolsDir}

Figure 3-81. Set Messages to Information - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.4.6.1 S12Z Assembler > Messages > Disable User Messages

Use this panel to specify the settings for disabling the user messages for the S12Z assembler to use.

The following table lists and describes the disable user messages options for S12Z assembler.

Table 3-298. Tool Settings - S12Z Burner > Messages > Disable User Messages Options

Option	Description
Disable all Messages	Use this option to disable all user messages. This option disables messages that are not in the normal message categories like, WARNING, INFORMATION, ERROR, or FATAL by reducing the amount of messages, and simplifying the error parsing of other tools.
Display Type of Messages	Use this option to disable the type of messages.
Disable Informal Messages (e.g. memory model, floating point format)	Use this option to disable the informal messages (e.g., memory model, floating point format).
Disable Included Files Messages	Use this option to disable the messages about the generated files.
Disable Reading Files Messages (e.g. input files)	Use this option to disable the messages about the reading files.
Disable Generated Files Messages	Use this option to disable the messages about the include files.
Disable Processing Statistics Messages (e.g. code size, RAM/ROM usage)	Use this option to disable the messages about processing statistics.

3.11.4.7 S12Z Assembler > General

Use this panel to specify the general assembler behavior.

The following table lists and describes the general assembler options for S12Z.

Table 3-299. Tool Settings - Assembler > General Options

Option	Description
MCUasm Compatibility	Check to activate the compatibility mode with the MCUasm Assembler.
Other Flags	Specify additional command line options for the assembler; type in custom flags that are not otherwise available in the UI.

3.11.5 S12Z Preprocessor

Use this panel to specify the preprocessor settings of the S12Z.

The following table lists and describes the preprocessor options for S12Z.

Table 3-300. Tool Settings - S12Z Preprocessor Options

Option	Description
Command	Shows the location of the preprocessor executable file. You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the assembler will be called with.
Expert Settings	Shows the command line pattern; default is `${COMMAND} -E ${FLAGS} ${INPUTS}.`
Command line pattern	

3.11.5.1 S12Z Preprocessor > Settings

Use this panel to specify the preprocessor settings of S12Z.

The following table lists and describes the preprocessor settings options for S12Z.

Table 3-301. Tool Settings - Preprocessor > Preprocessor Settings Options

Option	Description
Emit File/Line Breaks	Check to notify file breaks (or #line breaks) appear in the output.
Emit #pragma Directives	Check to show pragma directives in the preprocessor output. Essential for producing reproducible test cases for bug reports.
Emit #line Directives	Check to display file changes in comments (as before) or in `#line` directives.
Show Full Path	Check to display the full path in the preprocessor output.
Keep Comments	Check to display comments in the preprocessor output.
Keep Whitespace	Check to copy whitespaces in preprocessor output. This is useful for keeping the starting column aligned with the original source, though the compiler attempts to preserve space within the line. This does not apply when macros are expanded.

3.11.6 S12Z Disassembler

Use this panel to specify the command, options, and expert settings for S12Z Disassembler.

The following table lists and describes the disassembler options for S12Z.

Table 3-302. Tool Settings - DSC Disassembler

Option	Description
Command	Shows the location of the preprocessor executable file. Default value is `"${S12Z_ToolsDir}/decoder"` . You can specify additional command line options for the preprocessor; type in custom flags that are not otherwise available in the UI.
All options	Shows the actual command line the linker will be called with.
Expert settings	Shows the command line pattern; default is `${COMMAND} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`
Command line pattern	

3.11.6.1 S12Z Disassembler > Output

Use this panel to specify disassembler output settings.

The following table lists and describes the disassembler output settings options for S12Z.

Table 3-303. Tool Settings - S12Z Disassembler > Output

Option	Description
Print Full Listing	Prints a listing with the header information of the object file.
Write Disassembly Listing with Source File	Check to enable the decoder decoding Freescale object files write the source code within the disassembly listing. This option setting is default for the Freescale object files as input.
Decode DWARF Section	Check to write the DWARF section information in the listing file. Decoding from the DWARF section inserts this information in the listing file.
Configure Which Part of DWARF Information to Decode	Check to configure parts of DWARF information to decode.
Decode ELF Sections	Check to ensure that the ELF section information is also written to the listing file. Decoding from the ELF section inserts the following information in the listing file.
Dump ELF Section in LST File	Check to generate a HEX dump of all ELF sections in a LST file.

Table continues on the next page...

Table 3-303. Tool Settings - S12Z Disassembler > Output (continued)

Option	Description
Produce Inline Assembly File	Check to ensure that the output listing is an inline assembly file without additional information, but in C comments.
No Symbols in Disassembled Listing	Check to prevent symbols from printing in the disassembled listing.
Show Cycle Count for Each Instruction	Check to ensure that each instruction line contains the count of cycles in '[',']' braces. The cycle count is written before the mnemonics of the instruction. Note that the cycle count display is not supported for all architectures.
Write Disassembly Listing Only	Check to ensure that the Decoder decoding Freescale object files writes the source code within the disassembly listing only.
Write Disassembly Listing with Source and All Comments	Check to write the origin source and its comments within the disassembly listing.

3.11.6.2 S12Z Disassembler > Input

Use this panel to specify disassembler input settings.

The following table lists and describes the disassembler input settings options for S12Z.

Table 3-304. Tool Settings - S12Z Disassembler > Input

Option	Description
Object File Format	Use this option to specify the object file format. The options available are: • Automatic Detection (default) • ELF (DWARF 1.1/DWARF 2.0) • S-RECORD • Modula-2 Symbol File • Hex File • Binary File • HIWARE
Set processor	Specifies which processor should be decoded. For object files, libraries and applications, the processor is usually detected automatically. For S-Record and Intel Hex files, however, the decoder cannot determine which CPU the code is for, and therefore the processor must be specified with this option to get a disassembly output. Without this option, only the structure of a SRecord file is decoded.

3.11.6.3 S12Z Disassembler > Host

Use this panel to specify the host settings of the S12Z disassembler.

The following table lists and describes the host options for S12Z.

Table 3-305. Tool Settings - S12Z Disassembler > Host

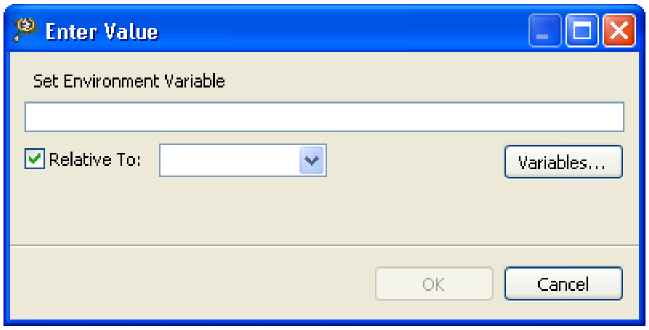
Option	Description
Borrow License Feature	This option allows you to borrow a license feature until a given date or time. Borrowing allows you to use a floating license even if disconnected from the floating license server.
Wait Until a License is Available from Floating License Server	By default, if a license is not available from the floating license server, then the application will immediately return. With -LicWait set, the application will wait (blocking) until a license is available from the floating license server.
Application Standard Occurrence	This option allows you to select the way you want the application window to start. Normally, the application starts with a normal window if no arguments are given. If you start the application with arguments (e.g., from the Maker to assemble, compile, or link a file), then the application runs minimized to allow for batch processing. However, you may specify the application's window behavior with the `View` option. • `-ViewWindow`, the application appears with its normal window. • `-ViewMin`, the application appears as an icon in the task bar. • `-ViewMax`, the application appears maximized (filling the whole screen). • `-ViewHidden`, the application processes arguments (e.g., files to be compiled or linked) in the background (no window or icon visible in the task bar).
Set Environment Variable	This option sets an environment variable. Use this environment variable in the maker, or use to overwrite system environment variables.

The following table lists and describes the toolbar buttons for the **Set Environment Variable** option.

Table 3-306. Toolbar Buttons - Set Environment Variable Option

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Environment Variable** option in the **S12Z Disassembler > Host** panel.



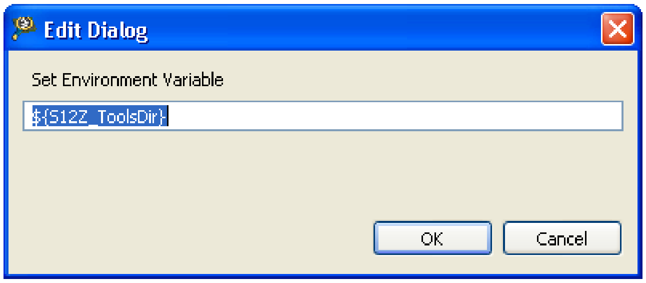
Enter Value

Set Environment Variable

☑ Relative To:

Figure 3-82. Set Environment Variable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Environment Variable** option in the **S12Z Disassembler > Host** panel.



Edit Dialog

Set Environment Variable

${S12Z_ToolsDir}

Figure 3-83. Set Environment Variable - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.6.4 S12Z Disassembler > Messages

Use this panel to specify whether to generate symbolic information for debugging.

The following table lists and describes the message options.

Table 3-307. Tool Settings - S12Z Disassembler > Messages Options

Option	Description
Don't Print INFORMATION Messages	This option allows you to disable the INFORMATION messsges. The `-W1` command inhibits the INFORMATION message reporting.
Don't Print INFORMATION or WARNING Messages	This option allows you to disable the printing of INFORMATION or WARNING messages. The `-W2` command suppresses all messages of the type INFORMATION or WARNING.
Create err.log Error File	You can use this option to enable the burner to create the err.log error file. The `-WErrFileOn` command creates or deletes the `err.log` file when the application is finished. When the errors occur, 16-bit window environments use the `err.log` files, containing a list of error numbers, to report the errors. If no errors occur, the 16-bit window environments delete the `err.log` file. By default, this checkbox is checked.
Create Error Listing File	You can use this option to create an error listing file. The `-WOutFileOn` command creates an error listing file. This option controls whether an error listing file should be created. The error listing file contains a list of all messages and errors that are created during processing. By default, this checkbox is checked.
Cut File Names in Microsoft Format to 8.3	This option truncates the filename to the 8.3 format. The `-Wmsg8x3` command cuts the filenames in Microsoft Format to 8.3. Some editors (early versions of WinEdit) expect the filename in Microsoft message format (8.3 format). That means the filename can have up to eight characters and no more than a three-character extension. Longer filenames are possible when you use Win95 or WinNT.
Set Message File Format for Batch Mode	Use this option to set the message file format for batch mode. The `-WmsgFb(-WmsgFbi,-WmsgFbm)` command sets the message file format for the batch mode. This option starts the Compiler with additional arguments (for example, files and Compiler options). If you start the Compiler with the arguments (for example, from the Make Tool or with the appropriate argument from an external editor), the Compiler compiles the files in a batch mode. No Compiler window is visible and the Compiler terminates after the job completion. The options available are: • Verbose Format • Microsoft Format (default)
Message Format for batch mode (e.g. %"%f%e%"(%l): %K %d: %m\n) (-WmsgFob)	This option modifies the default message format in batch mode. The `-WmsgFob` command sets the message format for batch mode. The supported formats are (assuming that the source file is `X:\Freescale\mysourcefile.cpph`): • `%s`: Source Extract • `%p`: Path (example, `X:\Freescale\`) • `%f`: Path and name (example, `X:\Freescale\mysourcefile`) • `%n`: filename (example, `mysourcefile`) • `%e`: Extension (example, `.cpph`) • `%N`: File (8 chars) (example, `mysource`)

Table continues on the next page...

Table 3-307. Tool Settings - S12Z Disassembler > Messages Options (continued)

Option	Description
	- `%E`: Extension (3 chars) (example, `.cpp`) - `%l`: Line (example, `3`) - `%c`: Column (example, `47`) - `%o`: Pos (example, `1234`) - `%K`: Uppercase kind (example, ERROR) - `%k`: Lowercase kind (example, error) - `%d`: Number (example, C1815) - `%m`: Message (example, text) - `%%`: Percent (example, %) - `\n`: New line - `%"`: A " if the filename, the path, or the extension contains a space - `%'`: A ' if the filename, the path, or the extension contains a space
Message Format for No File Info (e.g. %K %d: %m\n)	Use this option to set the message format for no file information. If there is no file information available for a message, then the `<string>` in the `-WmsgFonf<string>` command defines the message format string to use. The supported formats are: - `%K`: Uppercase kind (example, ERROR) - `%k`: Lowercase kind (example, error) - `%d`: Number (example, C1815) - `%m`: Message (example, text) - `%%`: Percent (example, %) - `\n`: New line - `%"`: A " if the filename, if the path or the extension contains a space - `%'`: A ' if the filename, the path or the extension contains a space
Message Format for No Position Info (e.g. %"%f%e%": %K %d: %m\n)	This option allows you to set the message format for no position information. If there is no position information available for a message, then the `<string>` in the `-WmsgFonp<string>` command defines the message format string to use. The supported formats are: - `%K`: Uppercase kind (example, ERROR) - `%k`: Lowercase kind (example, error) - `%d`: Number (example, C1815) - `%m`: Message (example, text) - `%%`: Percent (example, %) - `\n`: New line - `%"`: A " if the filename, if the path or the extension contains a space - `%'`: A ' if the filename, the path or the extension contains a space
Maximum Number of Error Messages	This option allows you to set the maximum number of error messages to be displayed. The `<number>` in the `-WmsgNe<number>` command sets the number of error messages to be displayed.
Maximum Number of Information Messages	This option allows you to set the amount of information messages that are logged. The `<number>` in the `-WmsgNi<number>` command specifies the maximum number of information messages allowed.

Table continues on the next page...

Table 3-307. Tool Settings - S12Z Disassembler > Messages Options (continued)

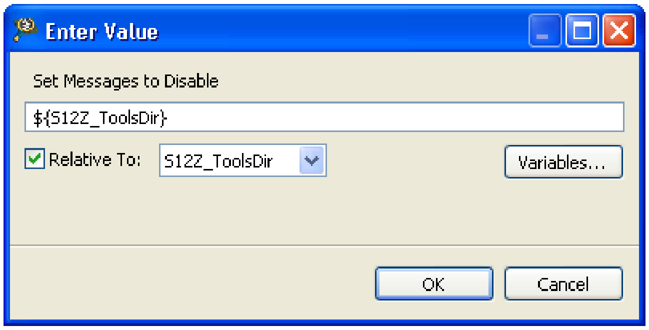
Option	Description
Set Messages to Disable	This option allows you to disable the specified messages. The `-WmsgSd<number>` command sets a message to disable, where `<number>` is the message number to be disabled, e.g., 1801.
Set Messages to Error	This option changes a message to an error message. The argument `<number>` in the command `-WmsgSe<number>` sets the specified message number to be an error, e.g., 1853.
Set Messages to Warning	This option sets a message to a warning message. The argument `<number>` of the `-WmsgSw<number>` command, sets the specified error number to be a warning, e.g., 2901.
Set Messages to Information	This option sets a message to an information message. The argument `<number>` of the command `-WmsgSi<number>` sets the specified message number to be an information, e.g., 1853.

The following table lists and describes the toolbar buttons for the **Set Messages to Disable** , **Set Messages to Error** , **Set Messages to Warning** and **Set Messages to Information** options of the **S12Z Disassembler > Messages** panel.

Table 3-308. Search Paths Toolbar Buttons - Messages Panel

Button	Description
➕	Add - Click to open the **Enter Value** dialog box and specify the object file search path.
✖	Delete - Click to delete the selected object file search path.
✎	Edit - Click to open the **Edit Dialog** dialog box and update the selected object file search path.
⇧	Move up - Click to move the selected object file search path one position higher in the list.
⇩	Move down - Click to move the selected object file search path one position lower in the list.

The following figure shows the **Enter Value** dialog box for the **Set Messages to Disable** option in the **S12Z Disassembler > Messages** panel.



Enter Value

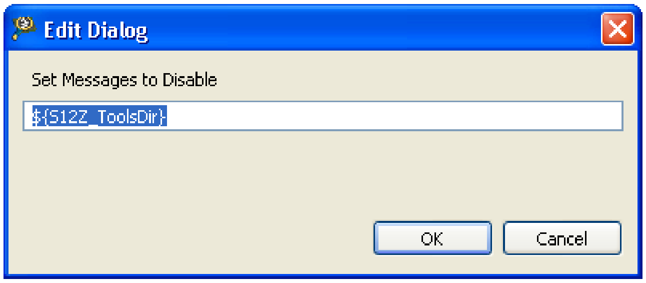
Set Messages to Disable

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-84. Set Messages to Disable - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Disable** option in the **S12Z Disassembler > Messages** panel.



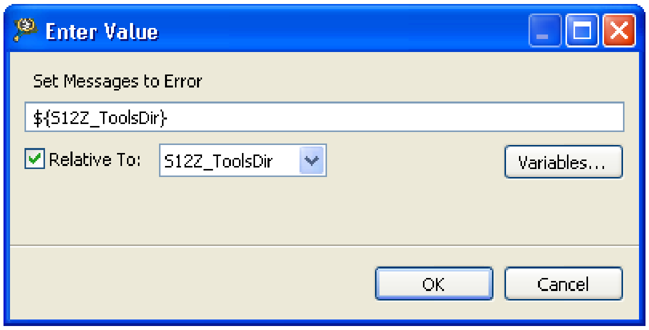
Edit Dialog

Set Messages to Disable

${S12Z_ToolsDir}

Figure 3-85. Set Messages to Disable - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Error** option in the **S12Z Disassembler > Messages** panel.



Enter Value

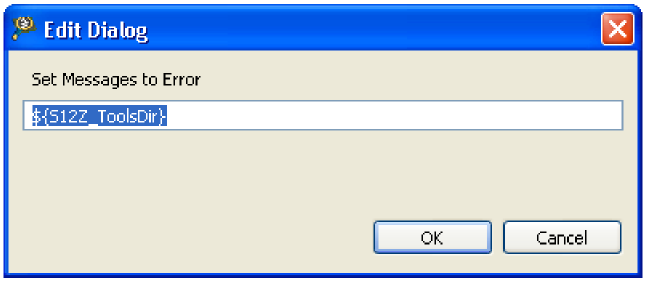
Set Messages to Error

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-86. Set Messages to Error - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Error** option in the **S12Z Disassembler > Messages** panel.

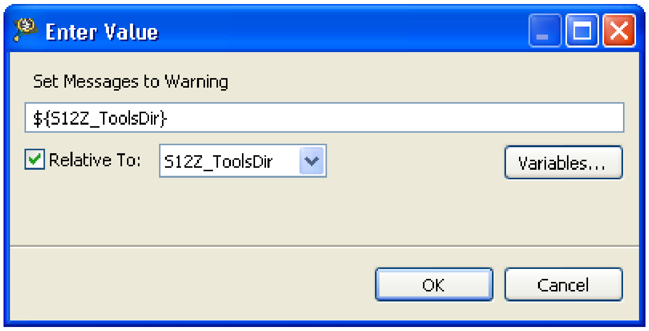


Set Messages to Error

${S12Z_ToolsDir}

Figure 3-87. Set Messages to Error - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Warning** option in the **S12Z Disassembler > Messages** panel.



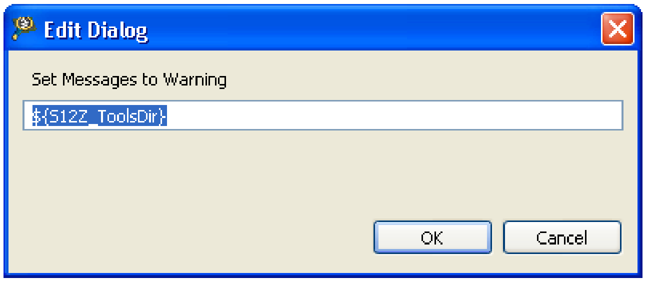
Set Messages to Warning

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-88. Set Messages to Warning - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Warning** option in the **S12Z Disassembler > Messages** panel.

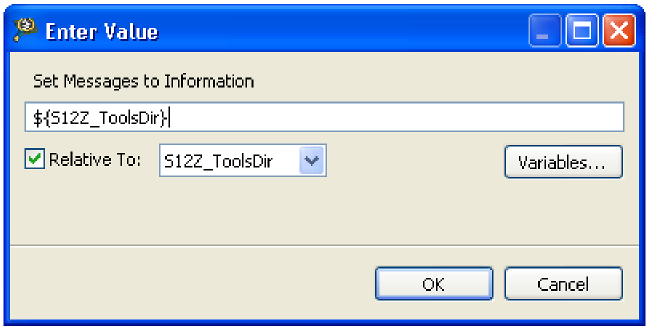


Set Messages to Warning

${S12Z_ToolsDir}

Figure 3-89. Set Messages to Warning - Edit Dialog

The following figure shows the **Enter Value** dialog box for the **Set Messages to Information** option in the **S12Z Disassembler > Messages** panel.



Enter Value

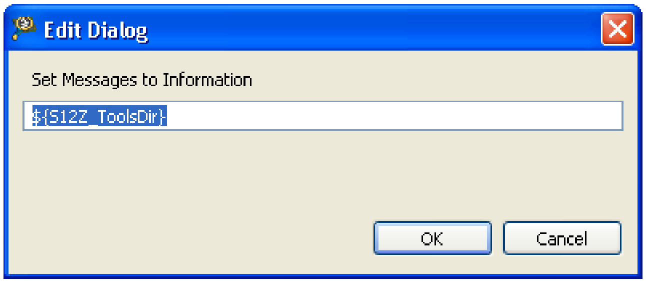
Set Messages to Information

${S12Z_ToolsDir}

☑ Relative To: S12Z_ToolsDir

Figure 3-90. Set Messages to Information - Enter Value Dialog Box

The following figure shows the **Edit Dialog** dialog box for the **Set Messages to Information** option in the **S12Z Disassembler > Messages** panel.



Edit Dialog

Set Messages to Information

${S12Z_ToolsDir}

Figure 3-91. Set Messages to Information - Edit Dialog

The buttons in the **Enter Value** and **Edit Dialog** dialog boxes help work with the object file search paths.

- Variables **-** Click to display the **Select build variable** dialog box and specify the object file search path variable. The resulting path variable, relative to the workspace, appears in the appropriate list.
- **OK-** Click to confirm the action and exit the dialog box.
- **Cancel-** Click to cancel the action and exit the dialog box.

3.11.6.4.1 S12Z Disassembler > Messages > Disable User Messages

Use this panel to specify the settings for disabling the user messages for the S12Z disassembler to use.

The following table lists and describes the disable user messages options for S12Z.

Table 3-309. Tool Settings - S12Z Disassembler > Messages > Disable User Messages

Option	Description
Disable all Messages	Use this option to disable all user messages. This option disables messages that are not in the normal message categories like, WARNING, INFORMATION, ERROR, or FATAL by reducing the amount of messages, and simplifying the error parsing of other tools.
Display Type of Messages	Use this option to disable the type of messages.
Disable Informal Messages (e.g. memory model, floating point format)	Use this option to disable the informal messages (e.g., memory model, floating point format).
Disable Included Files Messages	Use this option to disable the messages about the generated files.
Disable Reading Files Messages (e.g. input files)	Use this option to disable the messages about the reading files.
Disable Generated Files Messages	Use this option to disable the messages about the include files.
Disable Processing Statistics Messages (e.g. code size, RAM/ROM usage)	Use this option to disable the messages about processing statistics.

Chapter 4
Working with Debugger

A CodeWarrior project can have multiple associated launch configurations. A launch configuration is a named collection of settings that the CodeWarrior tools use.

The CodeWarrior project wizard generates launch configurations with names that follow the pattern projectname - configtype - targettype, where:

- projectname represents the name of the project
- configtype represents the type of launch configuration
- targettype represents the type of target software or hardware on which the launch configuration acts

If you use the CodeWarrior wizard to create a new project, the IDE creates two debugger related launch configurations:

- Debug configuration - Produces unoptimized code for development purposes.
- Release configuration - Produces code intended for production purposes.

The topics in this chapter are:

- Customizing Launch Configuration
- Debugging Bareboard Software
- Debugging Externally Built Executable Files

4.1 Customizing Launch Configuration

The **Debug Configurations** dialog box contains seven tabs allowing you to customize all aspects of a launch configuration.

NOTE

The CodeWarrior debugger shares some pages, such as Connection and Download. The settings that you specify in these pages also apply to the selected debugger.

NOTE

As you modify a launch configuration's debugger settings, you create pending, or unsaved, changes to that launch configuration. To save the pending changes, you must click the Apply button of the Debug Configurations dialog box, or click the Close button and then the Yes button.

NOTE

You can revert pending changes and restore their last saved settings. To undo pending changes, click the Revert button at the bottom of the Debug Configurations dialog box. The IDE restores the last set of saved settings to all pages of the Debug Configurations dialog box. Also, the IDE disables the **Revert** button until you make new pending changes.

The tabs in the **Debug Configurations** dialog box are:

- Main
- Arguments
- Debugger
- Source
- Environment
- Common
- Trace and Profile

4.1.1 Main

Use this page to specify the project and the application you want to run or debug.

The Main tab options are explained in the following table.

Table 4-1. Main Tab Options

Option	Description
Debug session type	Specifies the options to initiate a debug session using pre-configured debug configurations. The options include: • Download: Resets the target if the debug configuration specifies the action. Further, the command stops the target, (optionally) runs an initialization script, downloads the specified ELF file, and modifies the program counter(PC). • Connect: Runs the target initialization file specified in the RSE configuration to set up the board before connecting to it. The Connect debug session type does not load any symbolic debugging information for the current build target’s executable thereby, denying access to source-level debugging and variable display.

Table continues on the next page...

Table 4-1. Main Tab Options (continued)

Option	Description
	The Connect command resets the target if the launch configuration specifies this action. Further, the command stops the target, (optionally) runs an initialization script, does not load symbolics, download an ELF file, or modify the program counter(PC). NOTE: The default debugger configuration causes the debugger to cache symbolics between sessions. However, selecting the Connect option invalidates this cache. If you must preserve the contents of the symbolics cache, and you plan to use the Connect option, clear the Cache Symbolics Between Sessions check box in the Symbolics tab page. • Attach: Assumes that code is already running on the board and therefore does not run a target initialization file. The state of the running program is undisturbed. The debugger loads symbolic debugging information for the current build target's executable. The result is that you have the same source-level debugging facilities you have in a normal debug session (the ability to view source code and variables, and so on). The function does not reset the target, even if the launch configuration specifies this action. Further, the command loads symbolics, does not stop the target, run an initialization script, download an ELF file, or modify the program counter (PC). NOTE: The debugger does not support restarting debugging sessions that you start by attaching the debugger to a process. • Custom: Provides user an advantage to create a custom debug configuration
C/C++ application	• Project: Specifies the project to associate with the selected debug launch configuration. Click **Browse** to select a different project. • Application: Specifies the name of the C or C++ application. • Search Project: Click to open the **Program Selection** dialog box and select a binary. • Variables: Click to open the Select build variable dialog box and select the build variables to be associated with the program. **Note:** The dialog box displays an aggregation of multiple variable databases and not all these variables are suitable to be used from a build environment. Given below are the variables that should be used: • *ProjDirPath* - returns the absolute path of the current project location in the file system *${ProjDirPath}/Source/main.c*. • *workspace_loc* - returns the absolute path of a workspace resource in the file system, or the location of the workspace if no argument is specified *${workspace_loc:/ProjectName/Source main.c"${workspace_loc}* • *Gnu_Make_Install_Dir* - returns the absolute path of the GNU make.exe tool *${Gnu_Make_Install_Dir}\make.exe*

Table continues on the next page...

Table 4-1. Main Tab Options (continued)

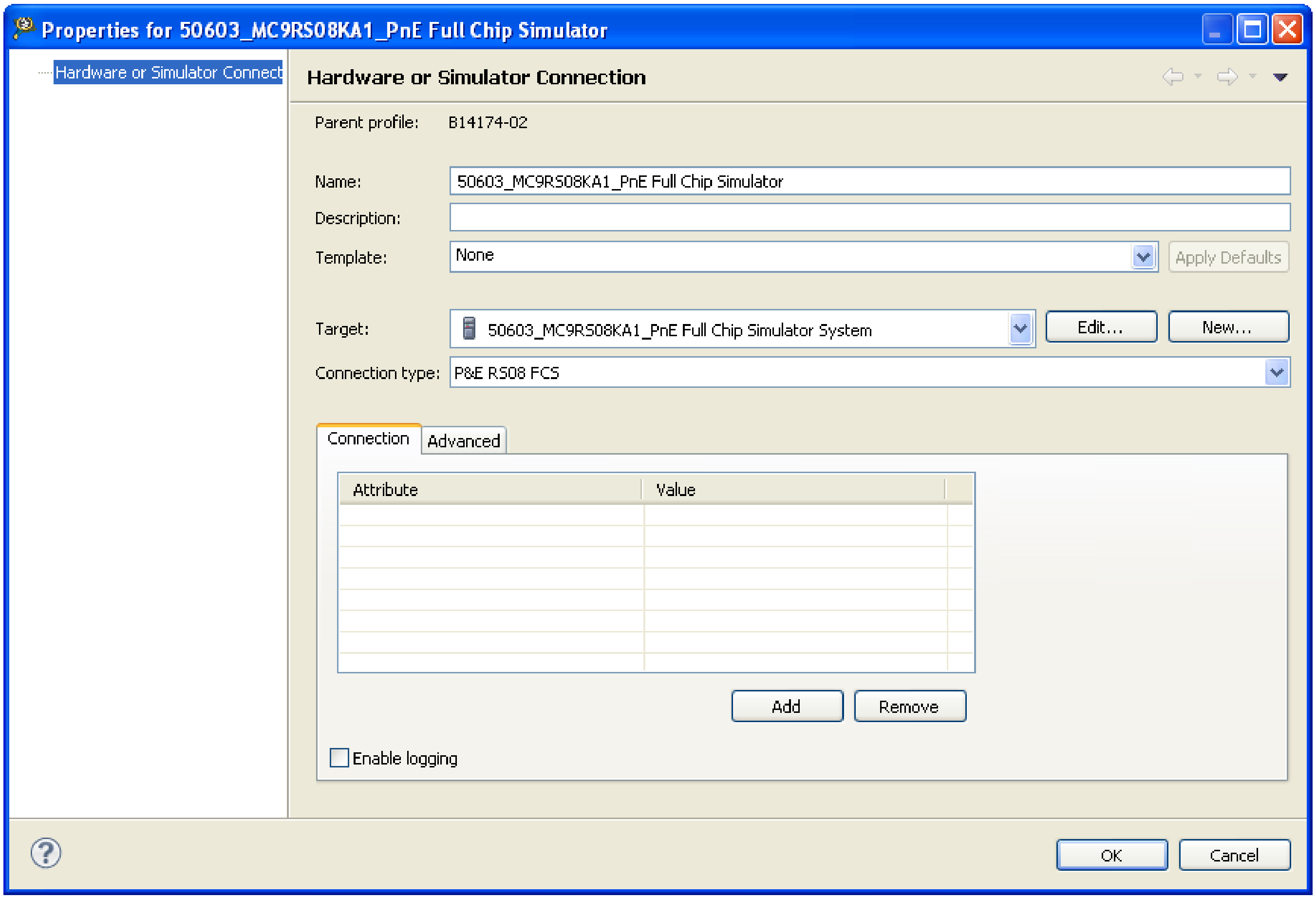
Option	Description
Build (if required) before launching	Controls how auto build is configured for the launch configuration. Changing this setting overrides the global workspace setting and can provide some speed improvements. NOTE: These options are set to default and collapsed when Connect debug session type is selected. The options include: • Build configuration – Specifies the build configuration either explicitly or use the current active configuration. • Select configuration using 'C/C++ Application' – Select/clear to enable/disable automatic selection of the configuration to be built, based on the path to the program. • Enable auto build – Enables auto build for the debug configuration which can slow down launch performance. • Disable auto build – Disables auto build for the debug configuration which may improve launch performance. No build action will be performed before starting the debug session. You have to rebuild the project manually. • Use workspace settings (default) – Uses the global auto build settings. • Configure Workspace Settings – Opens the Launching preference panel where you can change the workspace settings. It will affect all projects that do not have project specific settings.
Target settings	Specifies the connection and other settings for the target. The options include: • Connection – Specifies the applicable Remote System configuration. • Edit – Click to edit the selected Remote System configuration. • New – Click to create a new Remote System configuration for the selected project and application. • Execute reset sequence – Select to apply reset settings, specified in the target configuration, when attaching to a target. Alternatively, clear the option to ignore reset settings. NOTE: This option is not available when Connect debug session type is selected. • Execute initialization script(s) – Select to execute the initialization script(s), specified in the target configuration, when attaching to a target. Alternatively, clear the option to ignore the initialization script(s). NOTE: This option is not available when Connect debug session type is selected.

4.1.1.1 Editing Connection

To edit a remote system connection, click the **Edit** button in the **Main** tab of the **Debug Configurations** dialog box.

The **Properties for** <*project_connection*> dialog box appears. Here, <*project_connection*> is the name of the project followed by the associated connection. For example, *Proj_01_MC9S08AC128_PnE Full Chip Simulator*.

The remote system options in the **Properties for** <*project_connection*> dialog box change depending on the selected connection.


Properties for 50603_MC9RS08KA1_PnE Full Chip Simulator
Hardware or Simulator Connect
Hardware or Simulator Connection

Parent profile: B14174-02

Name: 50603_MC9RS08KA1_PnE Full Chip Simulator

Description:

Template: None

Target: 50603_MC9RS08KA1_PnE Full Chip Simulator System

Connection type: P&E RS08 FCS

Connection
Advanced
Attribute
Value
Add
Remove

□ Enable logging

Figure 4-1. Properties for <project_connection> Dialog Box

The following table lists the remote system options available in the **Properties for** <*project_connection*> dialog box.

Table 4-2. Properties for <project_connection> Dialog Box

Option	Description
Parent profile	Specifies the parent profile.
Name	Specifies the name of the connection used.
Template	Select the remote system template you want to use.
Edit	Click to edit the system type. **Note:** For more information on editing system types, refer to the topic Editing System Types in the chapter Multicore Debugging.
New	Click to create a new remote connection.

Table continues on the next page...

Table 4-2. Properties for <project_connection> Dialog Box (continued)

Option	Description
Connection type	Specifies the type of the connection in use. The list is populated based on the selected remote system type and system type. Note: The options in the **Connection** sub tab will change depending on the selected connection.
Connection	Enables you to specify connection settings for the selected connection type. For more information, refer to the topic Connection Tab Settings.
Initialization	Enables you to specify initialization settings for the selected connection type. For more information, refer to the topic Initialization Tab Settings.
System	Enables you to specify system settings for the selected connection type. For more information, refer to the topic System Tab Settings.
Advanced	Enables you to specify advanced settings for the selected connection type. For more information refer to the topic Advanced Tab Settings.
Buttons	• **OK** - Click to apply changes and close the **Properties for** <*project_connection*> dialog box. • **Cancel** - Click to close the **Properties for** <*project_connection*> dialog box without applying the changes.

4.1.1.2 Connection Tab Settings

Use this tab to specify the connection interface that the debugger uses to communicate with the connection on the target hardware. For more information on connections, refer to the respective connections chapter.

Table 4-3. Target Boards - Connection Types

Target Board	Connections
HCS08	Connections - HCS08 Architecture
RS08	Connections - RS08
ColdFire V1/ColdFire+ V1	Connections - ColdFire V1/ColdFire+ V1
ColdFire V2/3/4	Connections - ColdFire V2/3/4
Qorivva MPC55xx/56xx	Connections - Qorivva MPC55xx/56xx
Kinetis	Connections — Kinetis Architecture
DSC	Connections - DSC Architecture
S12Z	Connections - S12Z Architecture

4.1.1.3 Initialization Tab Settings

Enables you to specify initialization settings for the selected connection type.

Table 4-4. Initialization Settings

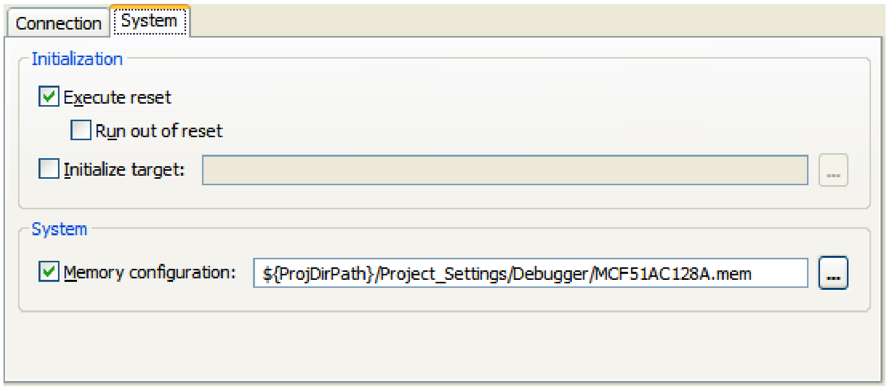
Option	Description
Initialize Target	Specifies the target initialization file to be used by the debugger at the start of each debugging session. Check this option to activate the **Target Initialization File** text box where you can specify the path of the initialization file. Alternatively, you can specify the file path by using any of the buttons listed below: • Workspace - Opens a dialog box where you can specify the initialization file in terms of a location relative to the IDE's workspace directory. After you select the file, the path to that file appears in the **Target Initialization File** text box, relative to the path of the variable `workspace_loc`. The IDE resolves this variable to the absolute file system path of the workspace directory root. • File System - Opens a dialog box where you can browse for the initialization file. After you select the file, the absolute path to that file appears in the **Target Initialization File** text box. • Variables - Opens a dialog box where you can specify the initialization file in terms of IDE path variables. After you specify the file, the path to that file appears in the **Target Initialization File** text box, relative to the path variables that you use. The IDE resolves each path variable as explained in the **Variable Description** box at the bottom of the Select Variable dialog box. Clear this option if you want the debugger to use a default target initialization file.
Use MemoryConfiguration File	Specifies the memory configuration file to be used by the debugger at the start of each debugging session. Check this option to activate the **Memory Configuration File** text box where you can specify the path of the configuration file. Alternatively, you can specify the file path by using any of the buttons listed below: • Workspace - Opens a dialog box where you can specify the initialization file in terms of a location relative to the IDE's workspace directory. After you specify the file, the path to that file appears in the **Memory Configuration File** text box, relative to the path of the variable `workspace_loc`. The IDE resolves this variable to the absolute file system path of the workspace directory root. • File System - Opens a dialog box where you can browse for the initialization file. After you specify the file, the absolute path to that file appears in the **Memory Configuration File** text box. • Variables - Opens a dialog box where you can specify the initialization file in terms of IDE path variables. After you specify the file, the path to that file appears in the **Memory Configuration File** text box, relative to the

Table 4-4. Initialization Settings

Option	Description
	path variables that you use. The IDE resolves each path variable as explained in the **Variable Description** box at the bottom of the Select Variable dialog box. Check the **Use Default** option to use the default memory configuration file and to deactivate the **Memory Configuration File** text box and the three buttons.

4.1.1.4 System Tab Settings

Enables you to specify system settings for the selected connection type.



☑ Execute reset

□ Run out of reset

□ Initialize target:

☑ Memory configuration: ${ProjDirPath}/Project_Settings/Debugger/MCF51AC128A.mem

Figure 4-2. System Tab Settings

The following table lists the available options in the **System** tab.

Table 4-5. System Settings

Option	Description
Initialization	
Execute Reset	Specifies that the debugger resets the target hardware before downloading a program for debugging purposes. Check this option to have the debugger reset the target before downloading the program to it. Clear this option to have the debugger download a program to the target without resetting that target.
Run out of reset	Determines what the ColdFire Microcontroller does after it is reset. Check this option to have the Microcontroller begin executing the program after it is reset. Clear this option to have the Microcontroller remain in debug mode after it is reset
Initialize target	Specifies the target initialization file to be used by the debugger at the start of each debugging session. Check this option to activate the adjacent text box where you can specify

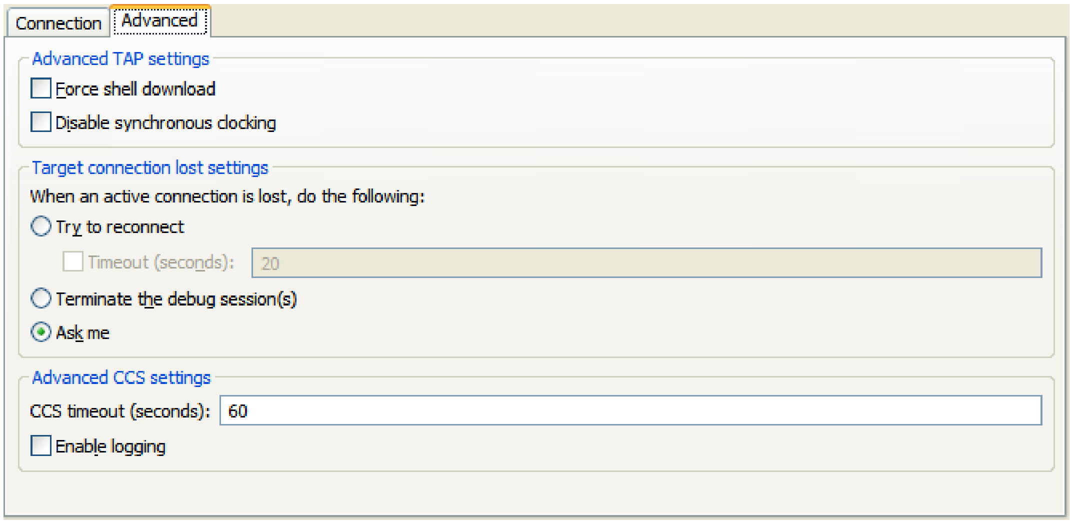
Table continues on the next page...

Table 4-5. System Settings (continued)

Option	Description
	the path of the initialization file. Alternatively, you can click **...** (Ellipsis) to open the **Initialize target** dialog box and specify the path of the file.
System	
Memory configuration	Specifies the memory configuration file to be used by the debugger at the start of each debugging session. Check this option to activate the adjacent text box where you can specify the path of the configuration file. Alternatively, you can click **...** (ellipsis) to open the **Memory configuration** dialog box and specify the path of the file.

4.1.1.5 Advanced Tab Settings

Enables you to specify advanced settings for the selected connection type.



Advanced TAP settings

□ Force shell download

□ Disable synchronous clocking

Target connection lost settings

When an active connection is lost, do the following:

○ Try to reconnect

□ Timeout (seconds): 20

○ Terminate the debug session(s)

● Ask me

Advanced CCS settings

CCS timeout (seconds): 60

□ Enable logging

Figure 4-3. Advanced Tab Settings

The following table lists the available options in the **Advanced** tab.

Table 4-6. Advanced Settings

Option	Description
Advanced TAP settings	
Disable synchronous clocking	Check this option to have the debugger use a standard (slow) procedure to write to memory on the target system. Clear this option to have the debugger use an optimized (fast) download procedure to write to memory on the target system. The fast download mechanism is used by default when writing to target memory. Check this option if the fast download procedure results in load failures.

Table continues on the next page...

Table 4-6. Advanced Settings (continued)

Option	Description
Target connection lost settings	
When an active connection is lost, do the following:	
Try to reconnect	Select if you want to reconnect to the target if the connection is lost.
Timeout (seconds)	Provide the time (in seconds) till you want to reconnect to the target.
Terminate the debug session(s)	Select if you want to terminate the debug section if the connection to target is lost.
Ask me	Select if you want the application to prompt you whether to terminate or reconnect if the target connection is lost.
Advanced CCS settings	
CCS timeout (seconds)	Enter the number of seconds after which you want the debugger to treat CCS as unresponsive. The debugger treats the time interval that you specify as a window of validity in which CCS must complete debugger requests. If CCS does not complete the requests during the specified time interval, the debugger treats CCS as unresponsive. For example, you might specify 30 seconds to give intensive CCS operations enough time to succeed during a debugging session. You do not want to wait for 30 seconds for the initial connect operation, if you mistyped the Ethernet TAP probe's IP address, or forgot to turn on the target hardware. For these reasons, the debugger treats the specified value differently for initial-connect operations.
Enable logging	Check this option to have the debugger output connection protocol activity to a console in the Console view. Clear this option if you do not want the debugger to output connection-protocol activity to a console.

4.1.2 Arguments

Use this page to specify the program arguments that an application uses and the working directory for a run or debug configuration.

The Arguments tab options are explained in the following table..

Table 4-7. Arguments Tab Options

Option	Description
Program arguments	Specifies the arguments passed on the command line.
Variables	Click to select variables by name to include in the program arguments list.
Working Directory	Specifies the run/debug configuration working directory.
Use default	Check to specify the local directory or uncheck to specify a different workspace, a file system location, or variable.

Table continues on the next page...

Table 4-7. Arguments Tab Options (continued)

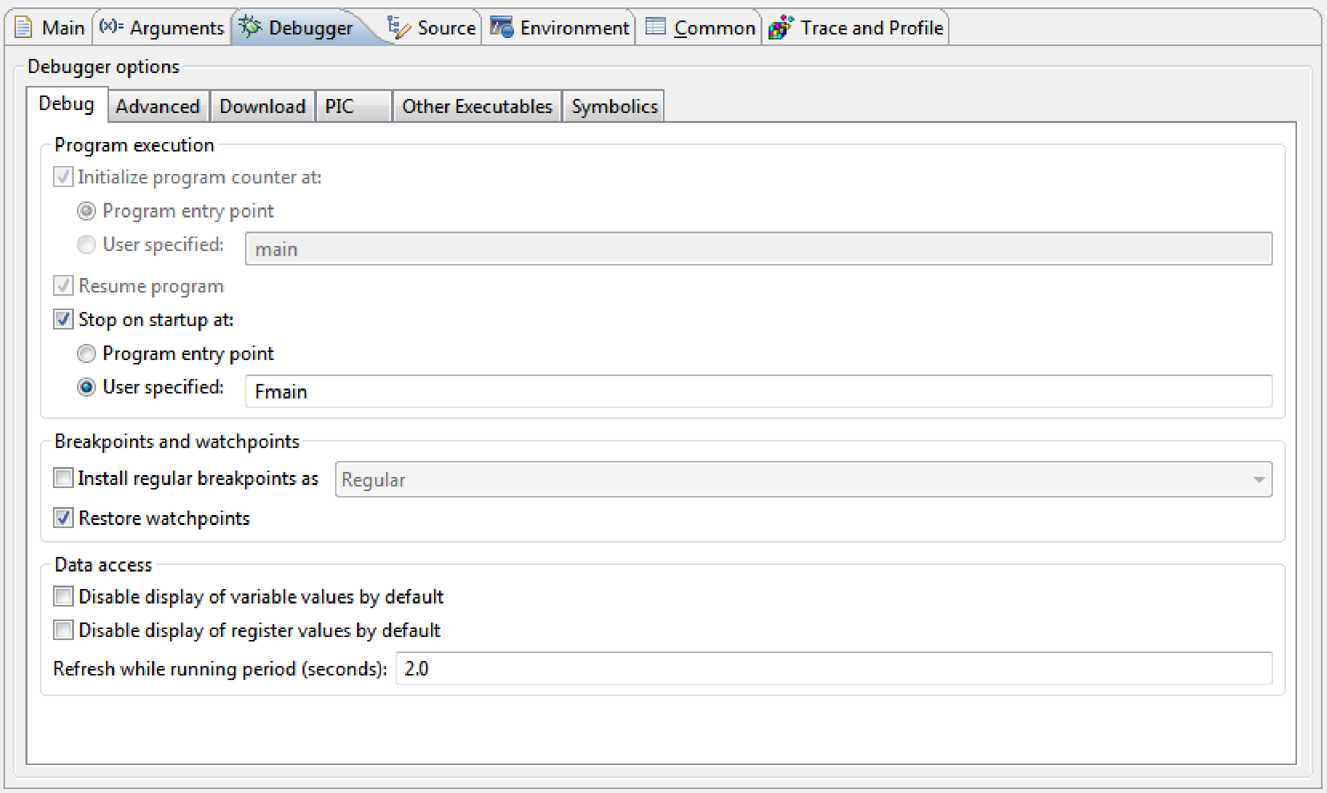
Option	Description
Workspace	Click to specify the path of, or browse to, a workspace relative working directory.
File System	Click to specify the path of, or browse to, a file system directory.
Variables	Click to specify variables by name to include in the working directory.

4.1.3 Debugger

Use this page to select a debugger to use when debugging an application. The **Debugger** tab presents different sub-tabs for specifying different settings.

NOTE

The sub-tabs under the **Debugger tab** change depending on the derivative and connection you select while creating the project.



Program execution

☑ Initialize program counter at:

● Program entry point

○ User specified: main

☑ Resume program

☑ Stop on startup at:

○ Program entry point

● User specified: Fmain

Breakpoints and watchpoints

□ Install regular breakpoints as Regular

☑ Restore watchpoints

Data access

□ Disable display of variable values by default

□ Disable display of register values by default

Refresh while running period (seconds): 2.0

Figure 4-4. Debug Configurations - Debugger Tab

NOTE

The subsequent topics explain the various settings that you can configure using the pages beneath the **Debugger** tab.

The sub-tabs available under the **Debugger** tab are:

- Debug
- Download
- PIC
- Other Executables
- Symbolics
- OS Awareness
- Exceptions
- Reset
- Interrupts
- Remote
- EPPC Exceptions
- System Call Services

4.1.3.1 Debug

Use this page to specify the program execution options, breakpoint, watchpoint options, and target access behavior.

The **Debug** tab options are explained in the following table.

Table 4-8. Debug Tab Options

Group	Option	Description
Program execution	Initialize program counter at	Initializes program at specified location.
	Program entry point	Select to initialize the debugger at a specified program entry point. Click **Advanced** to modify the default program entry points.
	User specified	Select to initialize the debugger at a user-specified function. The default location is `main`.
	Resume program	Check to enable program resume.
	Stop on startup at	Stops program at specified location. when unchecked, the program runs until you interrupt it manually, or until it hits a breakpoint.
	Program entry point	Select to stop the debugger at a specified program entry point. Click **Advanced** to modify the default program entry points.

Table continues on the next page...

Table 4-8. Debug Tab Options (continued)

Group	Option	Description
	User specified	Select to stop the debugger at a user-specified function. The default location is `main`.
Breakpoints and watchpoints	Install regular breakpoints as	Check this option to install breakpoints as either: • Regular, or • Hardware, or • Software Clear this option to install breakpoints as Regular breakpoints.
	Restore watchpoints	Check this option to restore previous watchpoints.
Data access	Disable display of variable values by default	Check this option to disable the display of variable values. Clear this option to enable the display of variable values.
	Disable display of register values by default	Check this option to disable the display of register values. Clear this option to enable the display of register values.
	Refresh while running period (seconds)	Specifies the refresh period used when a view is configured to refresh while the application is running.

4.1.3.2 Download

Use this tab to specify the program sections the debugger downloads to the target, and whether the debugger should read back those sections and verify them.

NOTE

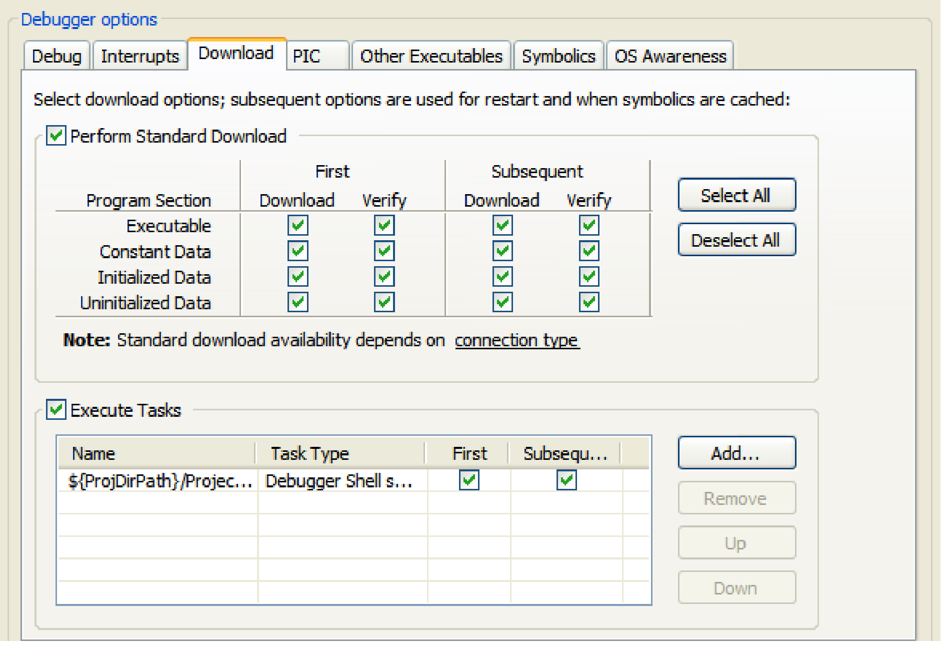
Checking all checkboxes in the **Program Download Options** group significantly increases download time.

The **Download** tab shows the Download tab. Briefly, the section data types are:

- **Executable** - These sections contain your program's code.
- **Constant Data** - These sections contain your program's constants. These values can not be modified.
- **Initialized Data** - Initialized data sections contain your program's modifiable data.
- **Uninitialized Data** - Uninitialized data sections contain your program's uninitialized variables.

The following table explains each option.

First options apply to the first debugging session. Subsequent options apply to successive debugging sessions. The **Download** options control whether the debugger downloads the specified section data type to the target hardware. The **Verify** options control whether the debugger reads the specified section data type from the target hardware and compares the data read against the data written to the device.



Select download options; subsequent options are used for restart and when symbolics are cached:

☑ Perform Standard Download

	First		Subsequent	
Program Section	Download	Verify	Download	Verify
Executable	☑	☑	☑	☑
Constant Data	☑	☑	☑	☑
Initialized Data	☑	☑	☑	☑
Uninitialized Data	☑	☑	☑	☑

Note: Standard download availability depends on connection type

☑ Execute Tasks

Name	Task Type	First	Subsequ...
${ProjDirPath}/Projec...	Debugger Shell s...	☑	☑

Figure 4-5. Download Tab

Table 4-9. Download Tab Settings

Section Data Type	Description
Executable	Controls downloading and verification for executable sections. Check appropriate checkboxes to specify downloading and verifications, for initial launch and for successive runs.
Constant Data	Controls downloading and verification for constant-data sections. Check appropriate checkboxes to specify downloading and verifications, for initial launch and for successive runs.
Initialized Data	Controls downloading and verification for initialized-data sections. Check appropriate checkboxes to specify downloading and verifications, for initial launch and for successive runs.
Uninitialized Data	Controls downloading and verification for uninitialized-data sections. Check appropriate checkboxes to specify downloading and verifications, for initial launch and for successive runs.
Select All	Selects all of the options available for downloading and verifying the program.
Deselect All	The debugger will not download or verify any program sections.
Execute Tasks	Check the option if you want to execute the listed tasks.

Table continues on the next page...

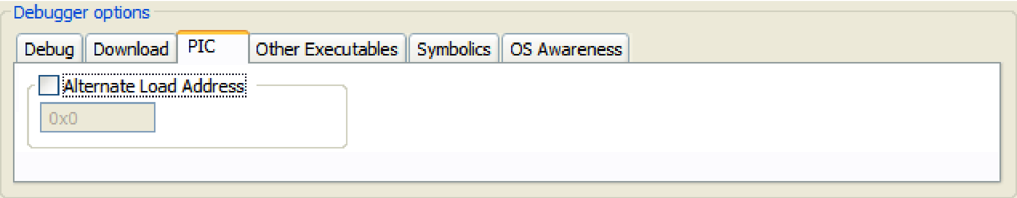
Table 4-9. Download Tab Settings (continued)

Section Data Type	Description
Add	Click to add download task(s).
Remove	Click to remove the listed download task(s).
Up	Click to move the listed task up in the priority list.
Down	Click to move the listed task down in the priority list.

4.1.3.3 PIC

Use this tab to specify an alternate address for the debugger to load a Position Independent Code (PIC) module on a target board. at an different address than specified in the ELF file. Also, when having to debug an application (such as U-Boot) built with ROM addresses after it has relocated itself to RAM. Usually, PIC is linked in such a way that the entire image starts at address 0x00000000. The PIC tab lets you specify an alternate address at which the debugger will load the PIC module in target memory.

The following figure shows the PIC tab.



□ Alternate Load Address

0x0

Figure 4-6. PIC Tab

Check the Alternate Load Address option and then enter the address (in hexadecimal notation) in the corresponding text box. The address that you specify is the starting address at which the debugger loads your program or finds it after runtime relocation. Specifying an alternate load address lets the debugger map the symbolic debugging information contained in the original ELF file to the relocated application image in RAM.

NOTE

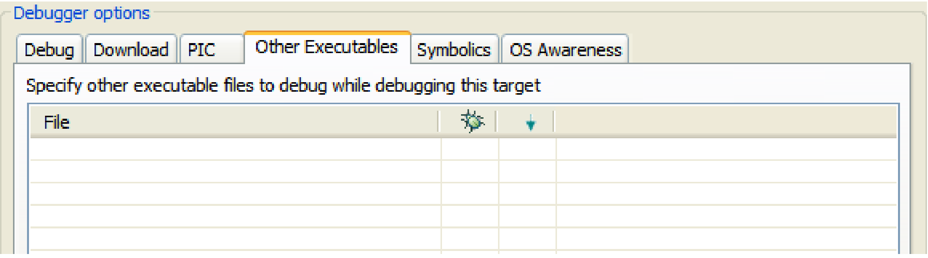
The debugger does not verify whether your code can execute at the specified address. As a result, the PIC generation settings of the compiler, linker and your program's startup routines must correctly set any base registers and perform any required relocations.

Clear the **Alternate Load Address** option to have the debugger load your program at a default starting address.

4.1.3.4 Other Executables

Use this tab to specify additional ELF files to download or debug in addition to the main executable file associated with the launch configuration.

The follow9ing figure shows the Other Executables tab view.



Specify other executable files to debug while debugging this target

Figure 4-7. Other Executables Tab

The following table describes the Other Executables debugger settings.

Table 4-10. Other Executables Tab Settings

Option	Description
File list	Shows files and projects that the debugger uses during each debug session The **Debug** column (🐞) - If this option is checked the debugger loads symbolics for the file. If you clear this option, the IDE does not load symbolics for the file. The **Download** column (↓) - If this option is checked the debugger downloads the file to the target device. If you clear this option, the debugger does not download the file to the target device.
Add	Click to open the Debug Other Executable dialog box. Use the dialog box to specify the following settings: • Specify the location of the additional executable - Enter the path to the executable file that the debugger controls in addition to the current project's executable file. Alternatively, click Browse to specify the file path. • Load symbols - Check this option to have the debugger load symbols for the specified file. Clear to prevent the debugger from loading the symbols. The **Debug** column of the **File** list corresponds to this setting. • Download to device - Check this option to have the debugger download the specified file to the target device. Clear this option to prevent the debugger from downloading the file to the device. The **Download** column of the **File** list corresponds to this setting. • OK - Click to add the information that you specify in the Debug Other Executable dialog box to the File list.

Table continues on the next page...

Table 4-10. Other Executables Tab Settings (continued)

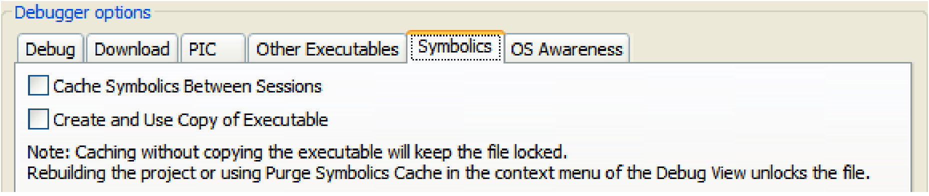
Option	Description
Change	Click to open the Debug Other Executable dialog box. The dialog box shows the current settings for the selected executable file in the **File** list column. Change this information as required and click OK to update the entry in the **File** list.
Remove	Click to remove the entry currently selected in the File list.

4.1.3.5 Symbolics

Use this tab to specify whether the debugger keeps symbolics in memory. Symbolics represent an application's debugging and symbolic information. Keeping symbolics in memory, known as caching symbolics, is beneficial when you debug a large-size application.

Consider a situation in which the debugger loads symbolics for a large application, but does not download content to a hardware device and the project uses custom makefiles with several build steps to generate this application. In such a situation, caching symbolics helps speed up the debugging process. The debugger uses the readily available cached symbolics during subsequent debugging sessions. Otherwise, the debugger spends significant time creating an in-memory representation of symbolics during subsequent debugging sessions.

The following figure shows the **Symbolics** tab.



☐ Cache Symbolics Between Sessions

☐ Create and Use Copy of Executable

Note: Caching without copying the executable will keep the file locked.
Rebuilding the project or using Purge Symbolics Cache in the context menu of the Debug View unlocks the file.

Figure 4-8. Symbolics Tab

The following table describes the Symbolics debugger settings.

Table 4-11. Symbolics Tab Settings

Option	Description
Cache SymbolicsBetween Sessions	Check this option to have the debugger cache symbolics between debugging sessions. With Create and Use Copy of Executable cleared, the executable file remains locked after the debugging session ends. In the Debug view, right-click the locked file and select Un-target Executables to have the debugger delete its symbolics cache and release the file lock. The IDE enables this menu command when there are

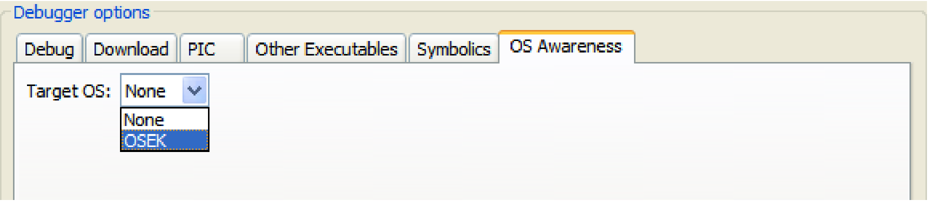
Table continues on the next page...

Table 4-11. Symbolics Tab Settings (continued)

Option	Description
	currently unused cached symbolics that it can purge. Clear this option so that the debugger does not cache symbolics between debugging sessions.
Create andUse Copy of Executable	Check this option to have the debugger create and use a copy of the executable file. Using the copy helps avoid file-locking issues with the build system. If you check this checkbox, the IDE can build the executable file in the background during a debugging session. Clear this option so that the debugger does not create and use a copy of the executable file.

4.1.3.6 OS Awareness

Use the **OS Awareness** tab to specify the operating system (OS) that resides on the target device.



Target OS: None

Figure 4-9. OS Awareness Tab - HCS08

Use the **Target OS** list box to specify the OS that runs on the target device, or specify None to have the debugger use the bareboard.

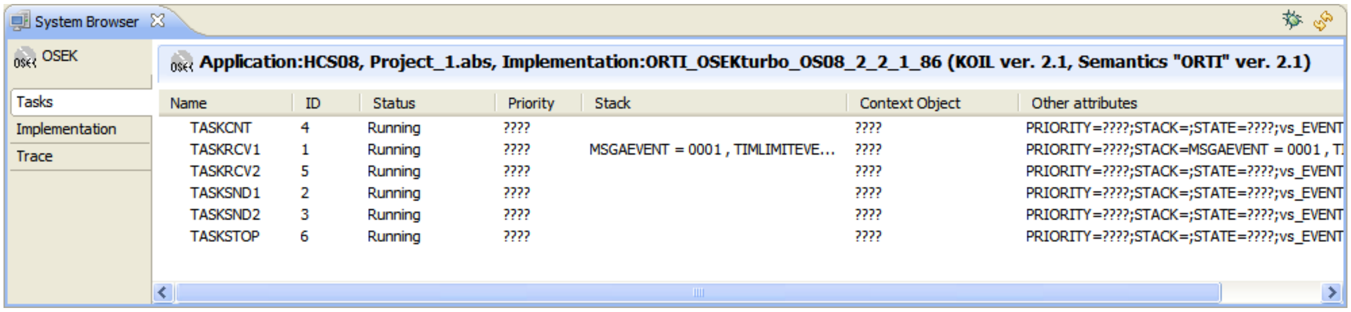
If you select OSEK as the target OS, you are required to specify the path of the OSEK Run Time Interface (ORTI) description file. You can click any of the following to navigate and browse to the ORTI file.

- Workspace - Click to open the **Folder Selection** dialog box and select a workspace location for the project. This is the directory that will contain the plug-ins and features to build, including any generated artifacts.
- File system - Click to open the **Browse For Folder** dialog box and select a folder.
- Variables - Click to open the **Select build variable** dialog box and select a variable to specify as an argument for the build directory, or create and configure simple build variables which you can reference in build configurations that support variables.

Selecting OSEK visualizes the RTOS's internal objects, like tasks, alarms, counters, and resources. Other objectives are:

- contains basic trace implementation that records and displays the last N values of the traceable attributes.
- enables multiple debuggers and handle multiple OSEK applications.
- enables components reuse.

Once the ORTI file is specified, click **Apply** and **Debug**. The ORTI file is automatically loaded at process creation time. Check the OSEK console to ensure whether the file was loaded or not. Any I/O exceptions or parse errors will be spilled out in the console. Once the ORTI file is successfully loaded, the **System Browser** view appears.



Application:HCS08, Project_1.abs, Implementation:ORTI_OSEKturbo_OS08_2_2_1_86 (KOIL ver. 2.1, Semantics "ORTI" ver. 2.1)

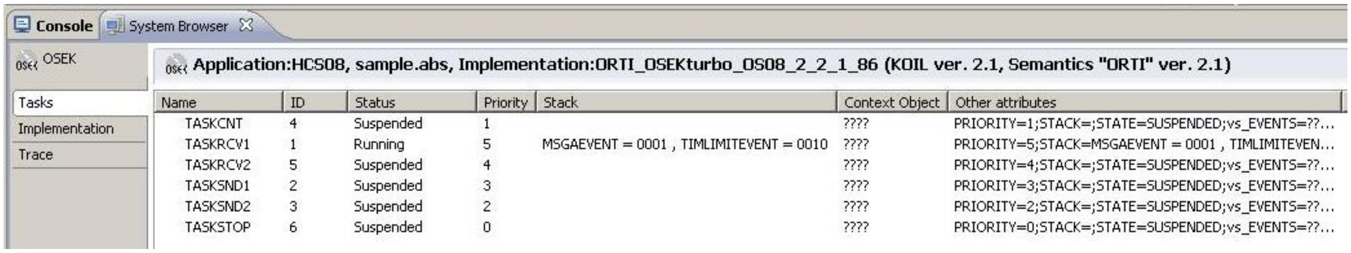
Name	ID	Status	Priority	Stack	Context Object	Other attributes
TASKCNT	4	Running	????		????	PRIORITY=????;STACK=;STATE=????;vs_EVENT
TASKRCV1	1	Running	????	MSGAEVENT = 0001 , TIMLIMITEVE...	????	PRIORITY=????;STACK=MSGAEVENT = 0001 , T
TASKRCV2	5	Running	????		????	PRIORITY=????;STACK=;STATE=????;vs_EVENT
TASKSND1	2	Running	????		????	PRIORITY=????;STACK=;STATE=????;vs_EVENT
TASKSND2	3	Running	????		????	PRIORITY=????;STACK=;STATE=????;vs_EVENT
TASKSTOP	6	Running	????		????	PRIORITY=????;STACK=;STATE=????;vs_EVENT

Figure 4-10. System Browser View

The three tabs in the **System Browser** view are: Tasks, Implementation, and Trace.

4.1.3.6.1 Tasks

The **Tasks** tab is a built-in view tab which lists only basic OS info like: task name, id, priority and state. For full information on application task open the **Implementation** tab.



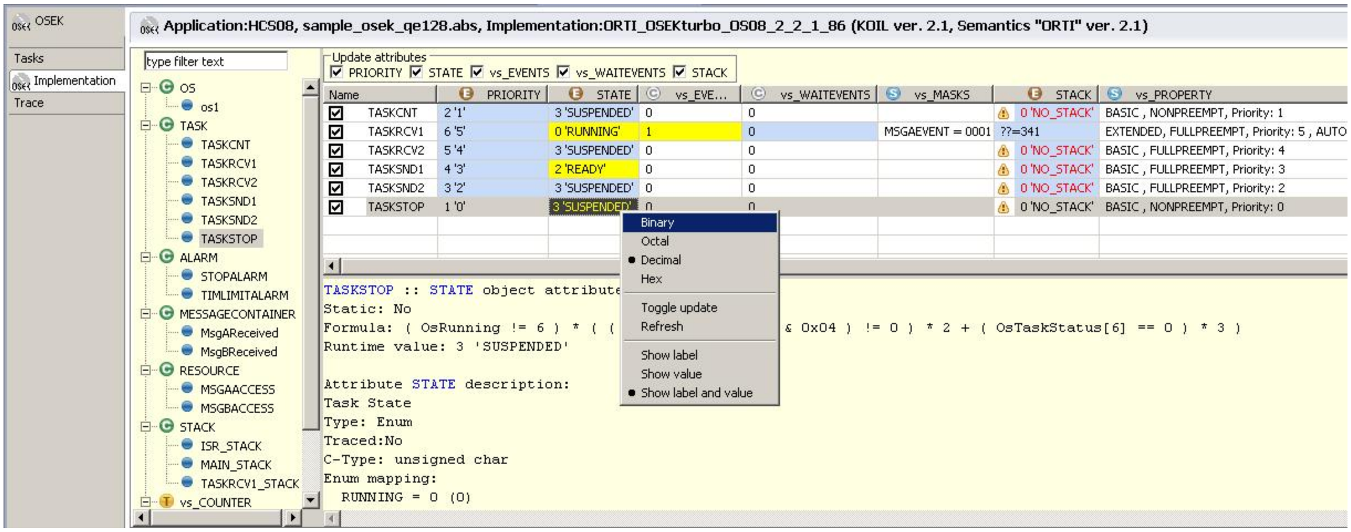
Application:HCS08, sample.abs, Implementation:ORTI_OSEKturbo_OS08_2_2_1_86 (KOIL ver. 2.1, Semantics "ORTI" ver. 2.1)

Name	ID	Status	Priority	Stack	Context Object	Other attributes
TASKCNT	4	Suspended	1		????	PRIORITY=1;STACK=;STATE=SUSPENDED;vs_EVENTS=??...
TASKRCV1	1	Running	5	MSGAEVENT = 0001 , TIMLIMITEVENT = 0010	????	PRIORITY=5;STACK=MSGAEVENT = 0001 , TIMLIMITEVEN...
TASKRCV2	5	Suspended	4		????	PRIORITY=4;STACK=;STATE=SUSPENDED;vs_EVENTS=??...
TASKSND1	2	Suspended	3		????	PRIORITY=3;STACK=;STATE=SUSPENDED;vs_EVENTS=??...
TASKSND2	3	Suspended	2		????	PRIORITY=2;STACK=;STATE=SUSPENDED;vs_EVENTS=??...
TASKSTOP	6	Suspended	0		????	PRIORITY=0;STACK=;STATE=SUSPENDED;vs_EVENTS=??...

Figure 4-11. System Browser View - Tasks

4.1.3.6.2 Implementation

The **Implementation** tab lists the full kernel object structure of the OSEK application and is composed of the Kernel objects tree and Kernel type viewer panels, refer topics Kernel Objects Tree Panel and Kernel Type Viewer Panel.



Application:HCS08, sample_osek_qe128.abs, Implementation:ORTI_OSEKturbo_OS08_2_2_1_86 (KOIL ver. 2.1, Semantics "ORTI" ver. 2.1)

Update attributes
☑ PRIORITY ☑ STATE ☑ vs_EVENTS ☑ vs_WAITEVENTS ☑ STACK

	Name	PRIORITY	STATE	vs_EVE...	vs_WAITEVENTS	vs_MASKS	STACK	vs_PROPERTY
☑	TASKCNT	2 '1'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , NONPREEMPT, Priority: 1
☑	TASKRCV1	6 '5'	0 'RUNNING'	1	0	MSGAEVENT = 0001	??=341	EXTENDED, FULLPREEMPT, Priority: 5 , AUTO
☑	TASKRCV2	5 '4'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , FULLPREEMPT, Priority: 4
☑	TASKSND1	4 '3'	2 'READY'	0	0		⚠ 0 'NO_STACK'	BASIC , FULLPREEMPT, Priority: 3
☑	TASKSND2	3 '2'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , FULLPREEMPT, Priority: 2
☑	TASKSTOP	1 '0'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , NONPREEMPT, Priority: 0

```
TASKSTOP :: STATE object attribute
Static: No
Formula: ( OsRunning != 6 ) * ( (                      & 0x04 ) != 0 ) * 2 + ( OsTaskStatus[6] == 0 ) * 3 )
Runtime value: 3 'SUSPENDED'

Attribute STATE description:
Task State
Type: Enum
Traced:No
C-Type: unsigned char
Enum mapping:
  RUNNING = 0 (0)
```

Figure 4-12. System Browser View - Implementation

4.1.3.6.3 Kernel Objects Tree Panel

Lists all kernel objects and their kernel types. To differentiate standard and vendor defined types a different icon is used. Each selection is reflected in the Kernel type viewer.

4.1.3.6.4 Kernel Type Viewer Panel

Lists all kernel objects of a certain kernel type. Each kernel object is described by its attributes.

- Update attributes panel - The role of this panel is to update kernel type attribute on each kernel object based on the next controller request. In UI terms, if an attribute is cleared the entire column is not renewed on the next update request. If the kernel type attribute is a String or the entire column contains only static attributes then the implementation attribute does not appear in this panel. Similarly, each table item have a checkbox which will prohibit the attribute update only for this kernel object.
- ORTI item description viewer. This is the lower text pane which displays static (or information extracted about that item from the ORTI file) and dynamic (runtime values) data. The ORTI description viewer also lists tags or links. A tag or link can be any ORTI item, that is any kernel object, type or attribute name. At load time

these names are put in global cache along with their ORTI entity. They are marked as hyperlinks and the open action will select that kernel type or object.

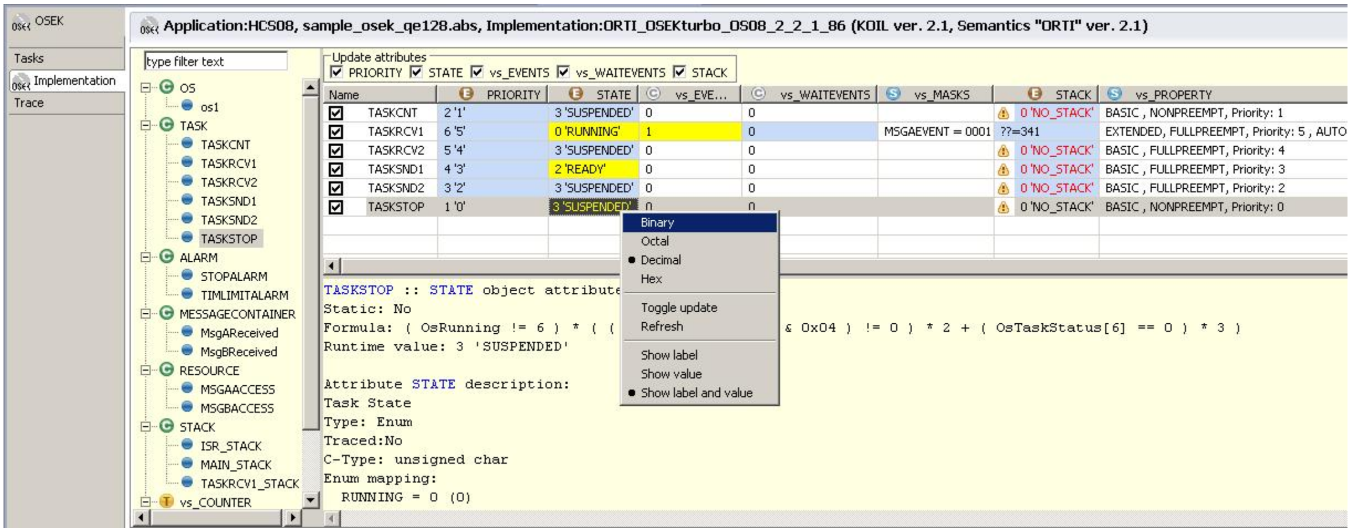
- Kernel objects detailed viewer - This table hold instances of the same kernel type. Intuitively, each table cell represents a kernel object attribute; except the first column which is allocated for object name. The table cells hold some graphical properties.

Table 4-12. Graphical Properties of Kernel Objects Detailed Viewer

Option	Description
Background Color	
White	Indicates static attributes like strings or constant C type or enum values; default.
Blue	Indicates dynamic attributes that need to be computed through C formulas or expressions; they are marked this way in order to distinguish them from static attributes.
Yellow	Indicates dynamic attributes to denote a value change.
Black	Indicates the table cursor or which cell is currently selected.
Foreground color	
Black	Indicates default text color
Red	Indicates an error or a value inconsistency.
Icons	Signifies the attribute type (Enum, String or C type) and the cells can a warning icon which indicates a possible error or an invalid value; table columns are appended with an image. For attribute SERVICETRACE the special step in and step out icons are used to indicate that the task has entered or left a service routine procedure.
Cell context menu	Contains options that change the attribute value representation and update commands. The first options are available only for integer attributes (not a String attribute) and they affect the attribute representation globally. This means that the change is saved into the model and all views will show that attribute consistently. A subgroup (Show label, Show value, Show label and value) of these options treat only enum attributes and handle label-value show. The other subgroup controls the base number representation (Binary, Octal, Decimal, Hex). The options from update group are: Toggle update which bypass the renew policies set to attribute column or row and Update which refreshes unconditionally the attribute's value direct from the target and not from model's.

4.1.3.6.5 Trace

This tab is designated to show a simple trace view of the traceable attributes. You can set the sample time and the trace buffer length. Because of a **System Browser** constraint this view illustrated trace results only when the core is stopped; for example. after a step.



Application:HCS08, sample_osek_qe128.abs, Implementation:ORTI_OSEKturbo_OS08_2_2_1_86 (KOIL ver. 2.1, Semantics "ORTI" ver. 2.1)

Update attributes
☑ PRIORITY ☑ STATE ☑ vs_EVENTS ☑ vs_WAITEVENTS ☑ STACK

Name	PRIORITY	STATE	vs_EVE...	vs_WAITEVENTS	vs_MASKS	STACK	vs_PROPERTY
☑ TASKCNT	2 '1'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , NONPREEMPT, Priority: 1
☑ TASKRCV1	6 '5'	0 'RUNNING'	1	0	MSGAEVENT = 0001	??=341	EXTENDED, FULLPREEMPT, Priority: 5 , AUTO
☑ TASKRCV2	5 '4'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , FULLPREEMPT, Priority: 4
☑ TASKSND1	4 '3'	2 'READY'	0	0		⚠ 0 'NO_STACK'	BASIC , FULLPREEMPT, Priority: 3
☑ TASKSND2	3 '2'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , FULLPREEMPT, Priority: 2
☑ TASKSTOP	1 '0'	3 'SUSPENDED'	0	0		⚠ 0 'NO_STACK'	BASIC , NONPREEMPT, Priority: 0

```
TASKSTOP :: STATE object attribute
Static: No
Formula: ( OsRunning != 6 ) * ( (                & 0x04 ) != 0 ) * 2 + ( OsTaskStatus[6] == 0 ) * 3 )
Runtime value: 3 'SUSPENDED'

Attribute STATE description:
Task State
Type: Enum
Traced:No
C-Type: unsigned char
Enum mapping:
  RUNNING = 0 (0)
```

Figure 4-13. System Browser View - Trace

4.1.3.7 Exceptions

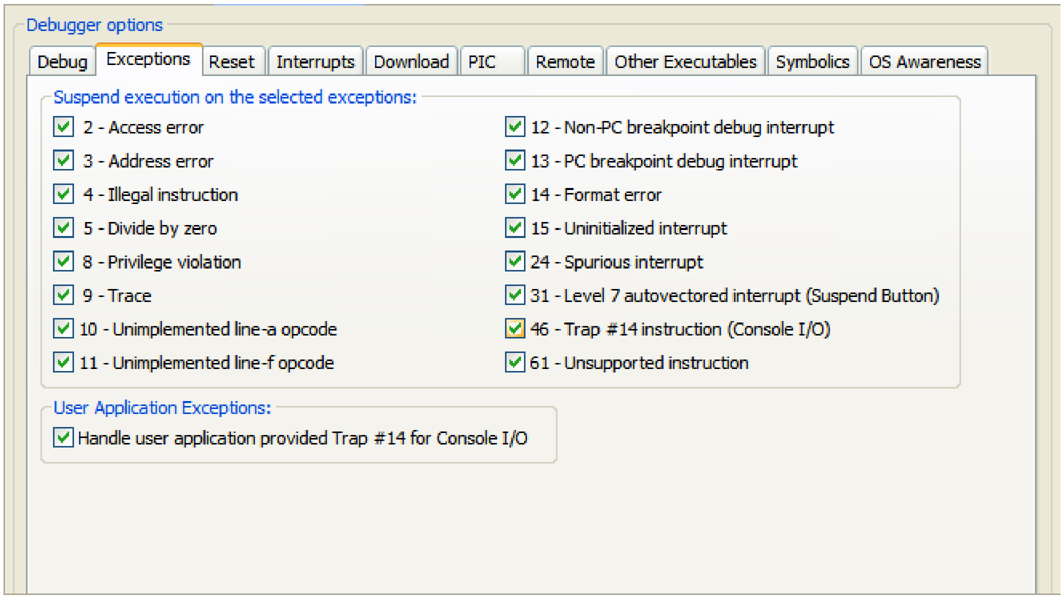
The **Exceptions** tab is available with P&E Microcomputer Systems, simulator, and Freescale USB and Ethernet TAP remote connections. Use this tab to specify hardware exceptions that you want the debugger to catch. Before you load and run the program, the debugger inserts its own exception vector for each exception you check in tab. To use your own exception vectors instead, clear the corresponding checkboxes.

If you check any options, the debugger reads the Vector_Based_Register (VBR), finds the corresponding existing exception vector and then writes a new vector at that register location. The address of this new vector is offset 0x408 from the VBR address. For example, if the VBR address is 0x0000 0000, the new vector at address 0x0000 0408 catches and handles the checked exceptions.

The debugger writes a Halt instruction and a Return from Exception instruction at this same location.

NOTE

If your exceptions are in Flash or ROM, do not check any boxes in the CF Exceptions panel.



Suspend execution on the selected exceptions:

- ☑ 2 - Access error
- ☑ 3 - Address error
- ☑ 4 - Illegal instruction
- ☑ 5 - Divide by zero
- ☑ 8 - Privilege violation
- ☑ 9 - Trace
- ☑ 10 - Unimplemented line-a opcode
- ☑ 11 - Unimplemented line-f opcode
- ☑ 12 - Non-PC breakpoint debug interrupt
- ☑ 13 - PC breakpoint debug interrupt
- ☑ 14 - Format error
- ☑ 15 - Uninitialized interrupt
- ☑ 24 - Spurious interrupt
- ☑ 31 - Level 7 autovectored interrupt (Suspend Button)
- ☑ 46 - Trap #14 instruction (Console I/O)
- ☑ 61 - Unsupported instruction

User Application Exceptions:

- ☑ Handle user application provided Trap #14 for Console I/O

Figure 4-14. Exceptions Tab - For ColdFire

The following table describes the exceptions settings.

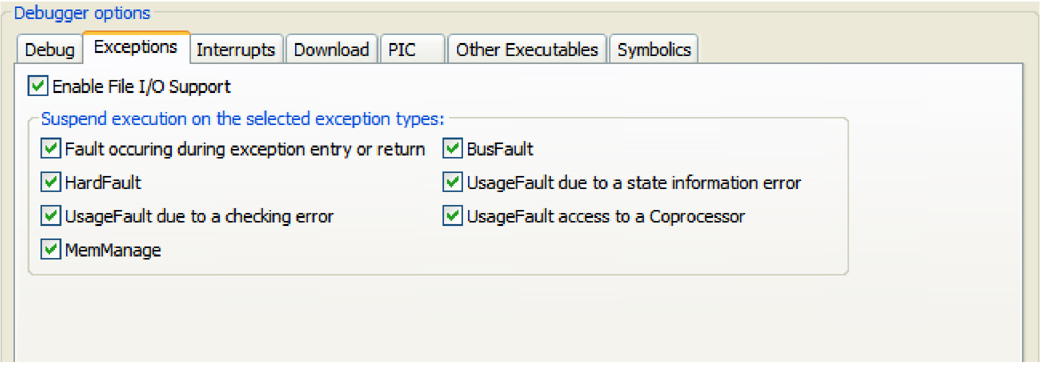
Table 4-13. Exceptions Settings - For ColdFire

Option	Description
2 - Access Error	Determines whether the debugger handles the access error exception. Check this option to catch and display access errors. Clear this option to ignore access errors.
3 - Address error	Determines whether the debugger handles the address error exception. Check this option to catch and display address errors. Clear this option to ignore address errors.
4 - Illegal instruction	Determines whether the debugger handles an invalid instruction exception. Check this option to catch and display invalid instructions. Clear this option to ignore invalid instructions.
5 - Divide by zero	Determines whether the debugger handles a divide by zero exception. Check this option to catch and display any attempt to divide by zero. Clear this option to ignore divide by zero attempts.
8 - Privilege violation	Determines whether the debugger handles a privilege violation exception. Check this option to catch and display privilege violations. Clear this option to ignore privilege violations.
9 - Trace	Determines whether the debugger handles a Trace exception. Check this option to catch and display trace exceptions. Clear this option to ignore trace exceptions.
10 -Unimplemented line-a opcode	Determines whether the debugger handles a unimplemented line-A opcode exception. Check this option to catch and display unimplemented line-A opcodes Clear this option to ignore unimplemented line-A opcodes.

Table continues on the next page...

Table 4-13. Exceptions Settings - For ColdFire (continued)

Option	Description
11- Unimplemented line-f opcode	Determines whether the debugger handles a unimplemented line-F opcode exception. Check this option to catch and display unimplemented line-F opcodes. Clear this option to ignore unimplemented line-F opcodes.
12 - Non-PC breakpoint debug interrupt	Determines whether the debugger handles non-PC breakpoint debug interrupts. Check this option to catch and display non-PC breakpoint debug interrupts. Clear this option to ignore non-PC breakpoint debug interrupts.
13 - PC breakpoint debug interrupt	Determines whether the debugger handles PC breakpoint debug interrupts. Check this option to catch and display PC breakpoint debug interrupts. Clear this option to ignore PC breakpoint debug interrupts.
14 - Format error	Determines whether the debugger handles format error exceptions. Check this option to catch and display format errors. Clear this option to ignore format errors.
15 - Uninitialized interrupt	Determines whether the debugger handles uninitialized interrupts. Check this option to catch and display uninitialized interrupts. Clear this option to ignore uninitialized interrupts.
24 - Spurious interrupt	Determines whether the debugger handles spurious interrupts. Check this option to catch and display spurious interrupts. Clear this option to ignore spurious interrupts.
31 - Level 7 autovectored interrupt (Suspend Button)	Determines whether the debugger handles level 7 suspend button exceptions. Check this option to catch and display the use of the level 7 interrupts. Clear this option to ignore level 7 interrupts.
46 - Trap #14 instruction (Console I/O)	Determines whether the debugger handles trap # 14 instructions, which implement console I/O. Clear this option to ignore trap 14 instructions. Check this option to catche and display uses of trap 14 instructions.
61 - Unsupported instruction	Determines whether the debugger handles the unsupported instruction exception. Check this options to catch and display unsupported instructions. Clear this option to ignore unsupported instructions.
Handle user application provided Trap #14 for console I/O	Determines whether the debugger handles trap # 14 exceptions when they occur in an application. Clear this option to ignore trap 14 instructions. Check this option to catches and display uses of trap 14 instructions.



Debugger options

☑ Enable File I/O Support

Suspend execution on the selected exception types:

☑ Fault occuring during exception entry or return ☑ BusFault

☑ HardFault ☑ UsageFault due to a state information error

☑ UsageFault due to a checking error ☑ UsageFault access to a Coprocessor

☑ MemManage

Figure 4-15. Exceptions Tab - For Kinetis

The following table describes the exceptions settings.

Table 4-14. Exceptions Settings - For Kinetis

Option	Description
Enable File I/O Support	Check to enable file input-output support.
Fault occuring during exception entry or return	Check to suspend execution if a fault occurs during exception or return.
Hard Fault	Generic fault that exists for all classes of fault that cannot be handled by any of the other exception mechanisms; typically used for unrecoverable system failure situations, though this is not required, and some uses of HardFault might be recoverable. HardFault is permanently enabled and has a fixed priority of -1.
UsageFault due to checking error	The UsageFault fault handles non-memory related faults caused by the instruction execution. A number of different situations will cause usage faults, including: • UNDEFINED Instructions • invalid state on instruction execution • errors on exception return • disabled or unavailable coprocessor access. The following can cause usage faults when the core is configured to report them: • unaligned addresses on word and halfword memory accesses • division by zero. UsageFault can be disabled (in this case, a UsageFault will escalate to HardFault). UsageFault has a configurable priority.
MemManage	MemManage fault handles memory protection related faults which are determined by the Memory Protection Unit or by fixed memory protection constraints, for both instruction and data generated memory transactions. The fault can be disabled (in this case, a MemManage fault will escalate to HardFault). MemManage has a configurable priority.
BusFault	Handles memory related faults other than those handled by the MemManage fault for both instruction and data generated memory transactions. These faults arise from errors detected on the system buses. Implementations are permitted to report

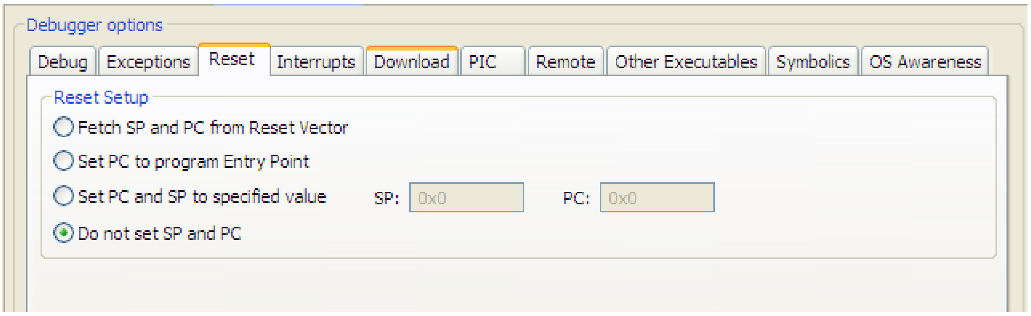
Table continues on the next page...

Table 4-14. Exceptions Settings - For Kinetis (continued)

Option	Description
	synchronous or asynchronous BusFaults according to the circumstances that trigger the exceptions. The fault can be disabled (in this case, a BusFault will escalate to HardFault). BusFault has a configurable priority.
UsageFault due to state information error	Check to suspend execution if a usage fault due to state information occurs.
UsageFault access to a Coprocessor	Check to suspend execution if a usage fault access to coprocessor occurs.

4.1.3.8 Reset

The **Reset** tab specifies the setup actions that the Microcontroller takes when it comes out of a reset.



Reset Setup

○ Fetch SP and PC from Reset Vector

○ Set PC to program Entry Point

○ Set PC and SP to specified value SP: 0x0 PC: 0x0

● Do not set SP and PC

Figure 4-16. Reset Tab

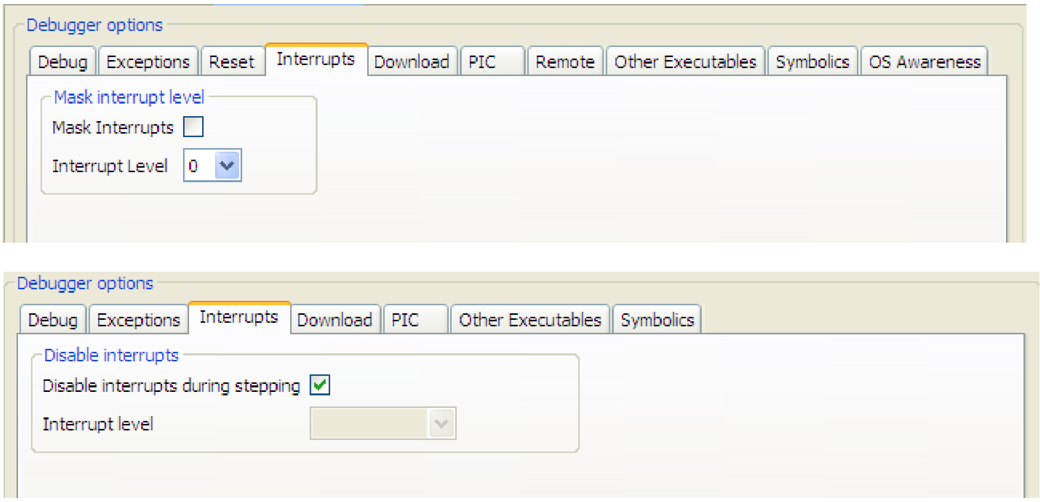
Table 4-15. Reset Tab

Option	Description
Fetch SP and PC from Reset Vector	When selected, the debugger fetches the base of the stack and the start address from the reset vector, and loads them into the Microcontroller's SP and PC, respectively. Used for ROM build targets.
Set PC to program Entry Point	When selected, the debugger sets the Microcontroller's PC to the program's start address. Used to emulate reset for RAM build targets.
Set PC and SP to specified value	When selected, the debugger takes the user-supplied values for SP and PC and loads them the corresponding Microcontroller registers. Used to reference the entry point of a boot loader. When selected, the SP: and PC: text entry boxes are active. Enter the hexadecimal addresses for SP and PC in these boxes.
Do not set SP or PC	When selected, the debugger takes no action and the Microcontroller uses the default addresses in the PC and SP.

4.1.3.9 Interrupts

Debugging an application involves single-stepping through code. However, if you do not modify the behavior of interrupts that are part of normal code execution, an interrupt may occur and the debugger jumps to the interrupt handler code, rather than single-stepping to the next instruction. Therefore, you must mask, or inhibit, certain interrupt levels to prevent the interrupts from happening. The interrupt levels that you inhibit varies, depending upon the microcontroller.

Use this tab to inhibit or allow interrupts. When inhibiting interrupts, you can mask interrupts below a level that you specify. The following figure shows the **Interrupts** tab.



Mask Interrupts ☐

Interrupt Level 0

Disable interrupts during stepping ☑

Interrupt level

Figure 4-17. Interrupts Tab

The following table explains each option.

Table 4-16. Interrupts Tab

Option	Description
Mask Interrupts	Determines whether the debugger inhibits or allows interrupts. Check this option to inhibit interrupts, using the level specified in the Interrupt Level option. Uncheck this option to permit all interrupts.
Interrupt Level	Use this option to specify the interrupt level that the debugger inhibits. Level 0 corresponds to the lowest priority interrupt, while level 7 is the highest. If you specify a level of 4, then the debugger inhibits interrupts of level 0 through 4, while interrupts at levels 5 through 7 execute.
For HCS08/RS08/Kinetis	
Disable interrupts during stepping	Check this checkbox if you want to disable interrupts during stepping.
Interrupt level	Use this option to specify the interrupt level.

NOTE

The exact definitions of interrupt levels are different for each target microcontroller, and masking all interrupts can cause erratic behavior. This means that finding the best interrupt level to mask can involve trial and error. Be alert for any code statements that change the interrupt mask; stepping over such a statement can modify the settings in the tab.

NOTE

The ARMv7-M profile supports two system level interrupts - PendSV for software generation of asynchronous system calls, and SysTick for a Timer integral to the ARMv7-M profile - along with up to 496 external interrupts. All interrupts have a configurable priority. The debugger can optionally set the C_MASKINTS bit in the DHCSR to inhibit (mask) PendSV, Systick, and external configurable interrupts from occurring during stepping. Where C_MASKINTS is set, permitted exception handlers which activate will execute along with the stepped instruction.

4.1.3.10 Remote

When debugging a Linux application, use this tab to specify where the debugger downloads the program for debug on the Linux host system, and whether to launch any optional applications while debugging.

The following figure shows the Remote tab view.

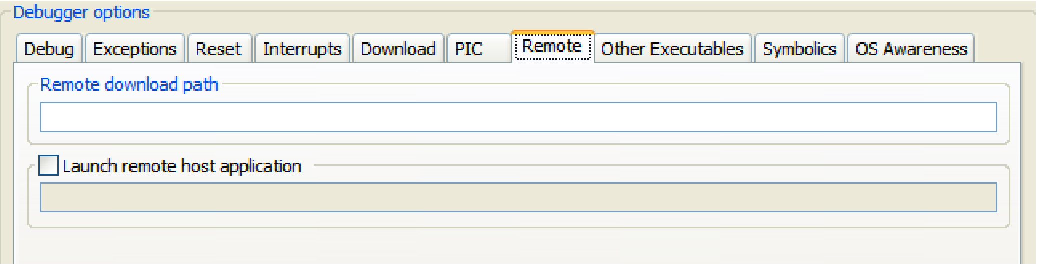


Figure 4-18. Remote Tab

The following table describes the Remote tab settings.

Table 4-17. Remote Tab Settings

Option	Description
Remote download path	Specifies the directory path that the debugger downloads the test program into.
Launch remote host application	Specifies the directory path of a Linux program that is to be launched along with the test program.

4.1.3.11 EPPC Exceptions

Use this page to specify which processor exceptions you want the debugger to catch.

The following table describes the EPPC Exceptions tab settings.

Table 4-18. EPPC Exceptions Tab Settings

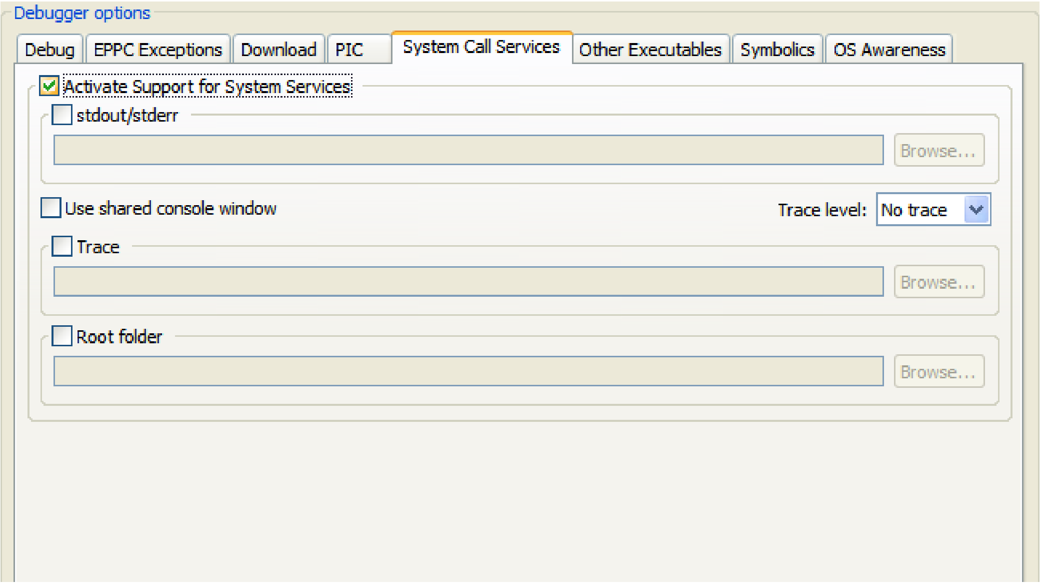
Option	Description
The features of this page view are currently not supported by this implementation.	

4.1.3.12 System Call Services

Use this page to activate the debugger's support for system calls and to select options that define how the debugger handles system calls. The CodeWarrior debugger provides system call support over JTAG. System call support lets bareboard applications use the functions of host OS service routines. This feature is useful if you do not have a board support package (BSP) for your target board. The host debugger implements these services. Therefore, the host OS service routines are available only when you are debugging a program on a target board or simulator.

NOTE

> The OS service routines provided must comply with an industry-accepted standard. The definitions of the system service functions provided are a subset of Single UNIX Specification (SUS).



☑ Activate Support for System Services

□ stdout/stderr

□ Use shared console window

Trace level: No trace

□ Trace

□ Root folder

Figure 4-19. Debug Configurations - System Call ServicesTab

The following table describes the settings on the System Call Services panel.

Table 4-19. System Call Services Settings

Option	Description
Activate Supportfor System Services	Check this option to enable support for system services. All the other options on the System Call Services panel are enabled only if you check this checkbox.
stdout/stderr	By default, the output written to stdout and stderr appears in a CodeWarrior IDE "console" window. To redirect console output to a file, check the stdout/stderr checkbox. Click Browse to display a dialog box and specify the path and name of this file.
Use sharedconsole window	Check this option if you wish to share the same console window between different debug targets. This setting is useful in multi-core or multi-target debugging.
Trace level	Use this dropdown list to specify the system call trace level. The place where the debugger displays the traced system service requests is determined by the Trace checkbox. The system call trace level options available are: • **No Trace** - system calls are not traced • **Summary** - the requests for system services are displayed • **Detailed** - the requests for system services are displayed along with the arguments/parameters of the request
Trace	By default, traced system service requests appear in a CodeWarrior IDE "console" window. To log traced system service requests to a file, check the Trace checkbox. Click Browse to display a dialog box and define the path and name of this file. In a project created by the New Project wizard, use the library syscall.a rather than a UART library for handling the output.

Table continues on the next page...

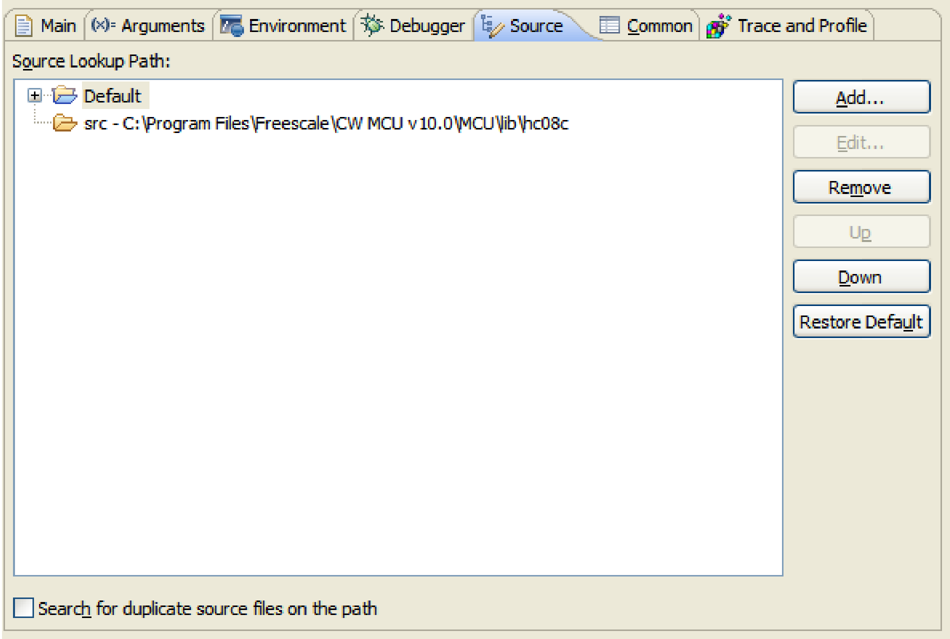
Table 4-19. System Call Services Settings (continued)

Option	Description
Root folder	The directory on the host system which contains the OS routines that the bareboard program uses for system calls.

4.1.4 Source

Use this page to specify the location of source files used when debugging a C or C++ application. By default, this information is taken from the build path of your project.

The Source tab options are explained in the following table..



Source Lookup Path:

- Default
- src - C:\Program Files\Freescale\CW MCU v10.0\MCU\lib\hc08c

□ Search for duplicate source files on the path

Figure 4-20. Debug Configurations - Source Tab

Table 4-20. Source Tab Options

Option	Description
Source Lookup Path	Lists the source paths used to load an image after connecting the debugger to the target.
Add	Click to add new source containers to the Source Lookup Path search list.
Edit	Click to modify the content of the selected source container.
Remove	Click to remove selected items from the **Source Lookup Path** list.

Table continues on the next page...

Table 4-20. Source Tab Options (continued)

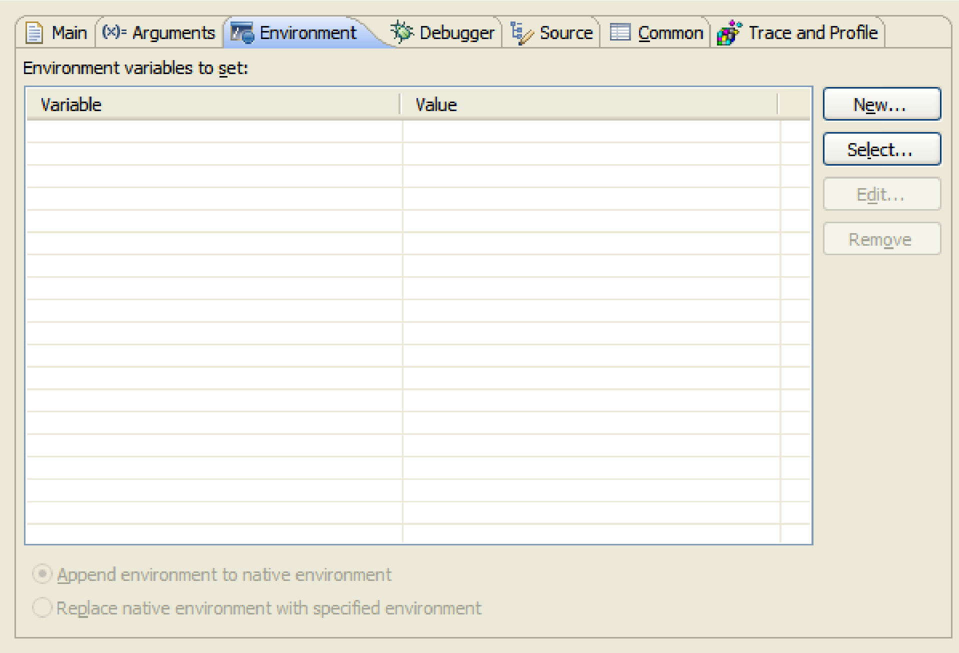
Option	Description
Up	Click to move selected items up the **Source Lookup Path** list.
Down	Click to move selected items down the **Source Lookup Path** list.
Restore Default	Click to restore the default source search list.
Search for duplicate source files on the path	Select to search for files with the same name on a selected path.

NOTE

For more information on path mapping refer to the topic *Path Mapping* in the *CodeWarrior Common Feature Guide*.

4.1.5 Environment

Use this page to specify the environment variables and values to use when an application runs.



● Append environment to native environment

○ Replace native environment with specified environment

Figure 4-21. Debug Configurations - Environment Tab

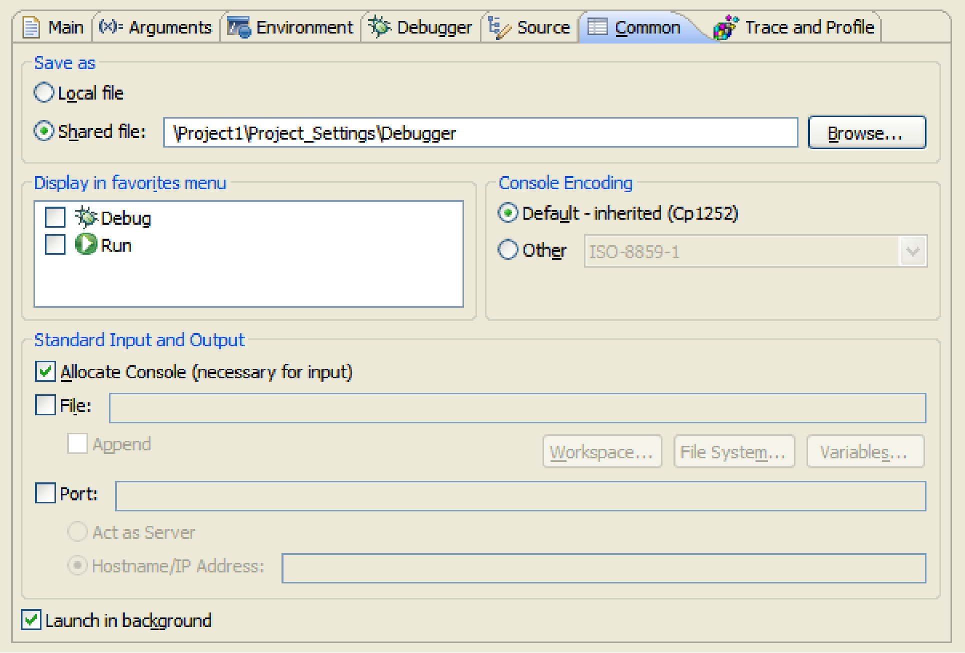
The following table list the **Environment** tab options.

Table 4-21. Environment tab options

Option	Description
Environment Variables to set	Lists the environment variable name and its value.
New	Click to create a new environment variable.
Select	Click to select an existing environment variable.
Edit	Click to modify the name and value of a selected environment variable.
Remove	Click to remove selected environment variables from the list.
Append environment to native environment	Select to append the listed environment variables to the current native environment.
Replace native environment with specified environment	Select to replace the current native environment with the specified environment set.

4.1.6 Common

Use this page to specify the location to store your run configuration, standard input and output, and background launch options.



Save as
○ Local file
● Shared file: \Project1\Project_Settings\Debugger

Display in favorites menu
□ Debug
□ Run

Console Encoding
● Default - inherited (Cp1252)
○ Other ISO-8859-1

Standard Input and Output
☑ Allocate Console (necessary for input)
□ File:
□ Append
□ Port:
○ Act as Server
● Hostname/IP Address:

☑ Launch in background

Figure 4-22. Debug Configurations - Common Tab

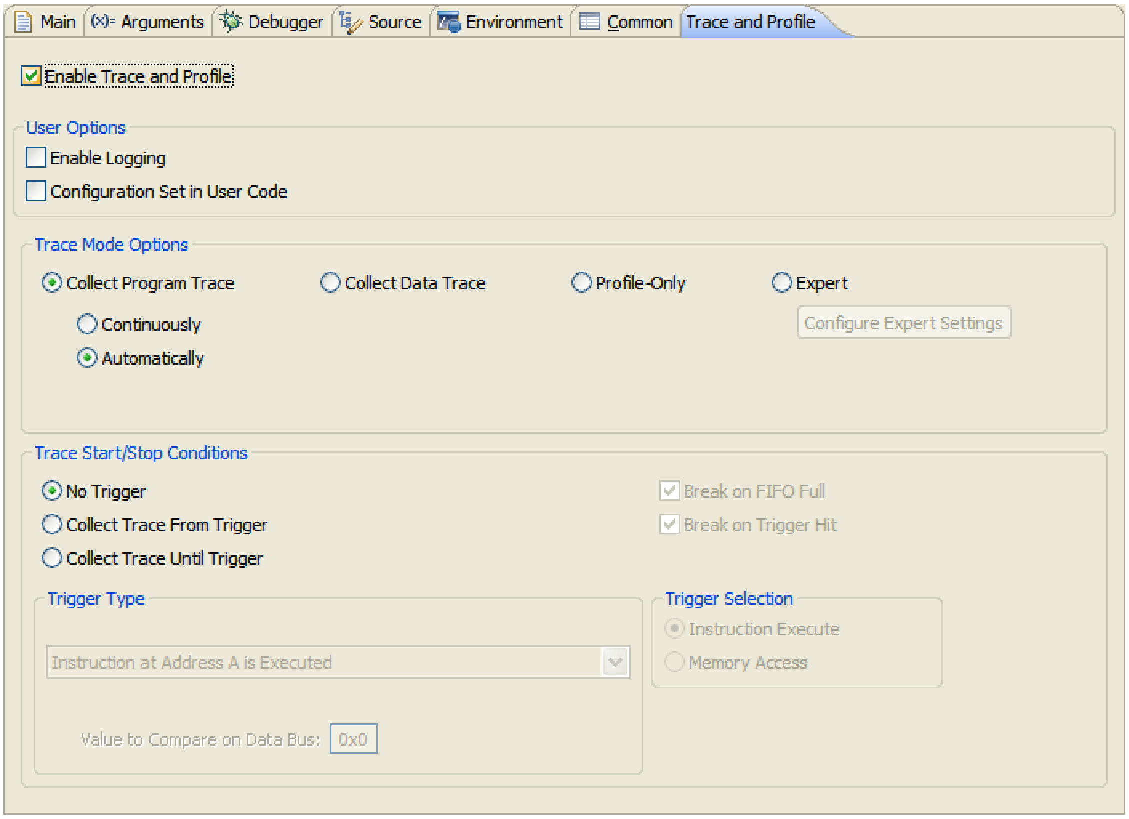
The following table lists and explains the the **Common** tab options.

Table 4-22. Common Tab Options

Option	Description
Save as	Select to save the launch configuration locally.
Local file	
Shared file	Select to specifies the path of, or browse to, a workspace to store the launch configuration file, and be able to commit it to a repository.
Display in favourites menu	Check to add the configuration name to Run or Debug menus for easy selection.
Encoding	Select an encoding scheme to use for console output.
Standard Input and Output	Select to assign a console view to receive the output.
Allocate Console (necessary for input)	
File	Specify the file name to save output.
Browse Workspace	Specifies the path of, or browse to, a workspace to store the output file.
Browse File System	Specifies the path of, or browse to, a file system directory to store the output file.
Variables	Select variables by name to include in the output file.
Append	Check to append output. Uncheck to recreate file each time.
Port	Check to redirect standard output (`stdout`, `stderr`) of a process being debugged to a user specified socket. **Note:** You can also use the `redirect` command in debugger shell to redirect standard output streams to a socket.
Act as Server	Select to redirect the output from the current process to a local server socket bound to the specified port.
Hostname/IP Address	Select to redirect the output from the current process to a server socket located on the specified host and bound to the specified port. The debugger will connect and write to this server socket via a client socket created on an ephemeral port
Launch in background	Check to launch configuration in background mode.

4.1.7 Trace and Profile

Use this page to configure the selected launch configuration for simulator and hardware profiling.



☑ Enable Trace and Profile

User Options

□ Enable Logging

□ Configuration Set in User Code

Trace Mode Options

● Collect Program Trace ○ Collect Data Trace ○ Profile-Only ○ Expert

○ Continuously

● Automatically

Trace Start/Stop Conditions

● No Trigger

○ Collect Trace From Trigger

○ Collect Trace Until Trigger

☑ Break on FIFO Full

☑ Break on Trigger Hit

Trigger Type

Instruction at Address A is Executed

Value to Compare on Data Bus: 0x0

Trigger Selection

● Instruction Execute

○ Memory Access

Figure 4-23. Debug Configurations - Trace and Profile Tab for HCS08

The following table describes the various Trace and Profile options for HCS08.

Table 4-23. Trace and Profile Options for HCS08

Group	Options	Descriptions
User Options	Enable Logging	When checked, creates a log file that keeps details of the actions that took place in the application. For example, when the debug session terminated, when the target execution resumed or stopped.
	Configuration Set in User Code	When checked, lets you configure trace registers from the application without using the Trace and Profile page. In this scenario, you can write the appropriate registers in the source code to configure the trace mode and triggers.
Trace Mode Options	Collect Program Trace	Consists of these options: • Continuously - When selected, collects the trace data continuously. The trace buffer is read, processed, and emptied periodically, so that the Trace Data viewer can collect all the trace records generated by the application. In this mode, the trace

Table continues on the next page...

Table 4-23. Trace and Profile Options for HCS08 (continued)

Group	Options	Descriptions
		data is not lost. It is a bit intrusive as it stops the target repeatedly in the background for collecting the trace buffers. • Automatically - When selected, the entries in the buffer start overwriting without interruption when the data reaches at the end of the buffer. If there is more trace data than the size of the buffer, the old entries will be overwritten. • LOOP1 Mode - Lets you collect the trace data without any consecutive identical addresses. If the next address to be stored in FIFO is the same as the one stored last time, it is ignored. This mode is particularly useful with short busy-wait type loops, which are repeated a large number of times or recursive calls, and is recommended when you want to view the coverage of that code, but not necessarily the number of times the code executed.For more information on Loop1 mode, refer to the MC9S08QE128 Reference Manual. Note: The LOOP1 Mode option is visible only for the debug version 3 (DbgVer 3) targets, that is HCS08 target with three comparators. For any other targets with two comparators, this option is not visible.
	Collect Data Trace	Collects the trace data of the values of a variable, which is located at the address where trigger B is set, for all the accesses (Read/Write/Both).
	Profile-Only	When selected, collects trace by sampling the program counter (PC) from time to time.
	Configure Expert Settings	When selected, enables the Configure Expert Settings button and gives you access to most of the on-chip DBG module registers. To configure expert settings, download the processor specific manual from the site: http://www.freescale.com/
Trace Start/Stop Conditions	No Trigger	Specifies that no triggers are set for collecting trace. When no triggers are set and trace is collected, the trace data starts collecting from the beginning of the application.

Table continues on the next page...

Table 4-23. Trace and Profile Options for HCS08 (continued)

Group	Options	Descriptions
	Collect Trace From Trigger	Starts collecting trace when the triggers generate, that is when the condition for A and B is met.
	Break on FIFO Full	While debugging, suspends the application automatically when buffer gets full. The checkbox gets enabled in the Automatically mode when the Collect Trace From Trigger option is selected.
	Collect Trace Until Trigger	Starts collecting trace and stops when the condition for triggers, A and B is met. This option is not enabled in the Continuously mode.
	Break on Trigger Hit	While debugging, suspends the application automatically when the trigger is hit, that is when the trigger condition is met. The checkbox gets enabled when the Collect Trace Until Trigger option is selected.
Trigger Type		Contains various conditions of triggers, A and B for starting/stopping trace collection.
	Instruction at Address A is Executed	Starts trace from the address or source line corresponding to trigger A.
	Instruction at Address A or Address B is Executed	Starts trace from the address or source line corresponding to trigger A or trigger B whichever occurs first.
	Instruction Inside Range from Address A to Address B is Executed	Starts trace when any instruction in the range between trigger address A and trigger address B is executed. That is, when *[address at trigger A] <= [current address] <= [address at trigger B].*
	Instruction Outside Range from Address A to Address B is Executed	Starts trace when any instruction outside the range between trigger address A and trigger address B is executed. That is, when *[current address] < [address at trigger A or address at trigger B] < [current address].*
	Instruction at Address A, Then Instruction at Address B are Executed	Starts trace from trigger B only if trigger A occurred before.
	Instruction at Address A is Executed and Value on Data Bus Match	Collects the trace data from the instruction where trigger A is set when the value specified in the **Value to Compare on Data Bus** text box matches with the opcode read from trigger A address, that is the value in memory at trigger A address. Note: Because the hardware has a small delay in enabling the triggers, trace won't be collected as expected if data match is done for the instruction immediately following the line where trigger is set.

Table continues on the next page...

Table 4-23. Trace and Profile Options for HCS08 (continued)

Group	Options	Descriptions
	Instruction at Address A is Executed and Value on Data Bus Mismatch	Collects the trace data from the instruction, where trigger A is set, on data mismatch. That is, trace is triggered at address A when the value specified in the **Value to Compare on Data Bus** text box does not match with the opcode read from trigger A address.
	Value to Compare on Data Bus	Contains the value that you specify to be matched or not matched with the opcode read from trigger A address.
	Capture Read/Write Values at Address B	Captures accesses to the variable address, where trigger B is set, after you press resume. Appears only when the **Collect Data Trace** mode is selected.
	Capture Read/Write Values at Address B, After Access at Address A	Waits for the program to execute the instruction at the address where trigger A is set, monitors the variable address where trigger B is set, and collects trace from there. Appears only when the **Collect Data Trace** mode is selected.
Trigger Selection	Instruction Execute	This option is related to how the hardware executes triggering. An address is triggered only when the opcode is actually executed, but this circuitry has a delay which sometimes makes the very next instruction in memory not caught in the trace when you press resume. In this mode, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions.
	Memory Access	When selected, allows memory access to both variables and instructions. Refer to the Memory Triggers topic in the Profiling and Analysis User Guide.

NOTE

For Memory trigger and other details with respect to setting tracepoints, refer to the Profiling and Analysis User Guide.

The following table describes the various Trace and Profile options for ColdFire V1.

Table 4-24. Trace and Profile Options for ColdFire V1

Group	Options	Descriptions
User Options	Enable Logging	Creates a log file that keeps details of the actions that took place in the application. For example, when the debug session terminated, when the target execution resumed or stopped.

Table continues on the next page...

Table 4-24. Trace and Profile Options for ColdFire V1 (continued)

Group	Options	Descriptions
	Configuration Set in User Code	When checked, lets you configure trace registers from the application without using the Trace and Profile page. In this scenario, you can write the appropriate registers in the source code to configure the trace mode and triggers.
Select Trace Mode	Continuous	When selected, collects the trace data continuously. It produces best possible trace and profile results because it captures all executed instructions. However, it is slow and intrusive as it stops the target in the background every about `500` cycles.
	Automatic (One-buffer)	When selected, captures only the last instructions executed before the target gets suspended. It is totally unintrusive.
	Halt the Target when Trace Buffer Gets Full	Appears only when the Automatic (One-Buffer) option is selected. It acts as a breakpoint for stopping the application. If selected, stops the application automatically when trace buffer gets full.
	Profile-Only. Sample PC every cycles	When selected, captures the PC address every `N` cycles, where `N` is `128/256/512.......16384`. Trace is mostly irrelevant in this mode, but Profile Statistics will be fairly accurate for a long cyclic run. This method is a bit intrusive because it stops the target in the background every about `8*N` cycles.
	Expert	When selected, enables the Configure Expert Settings button and lets you configure the ColdFire V1 trace and debug registers directly.
Trace Start/Stop Conditions		Includes various conditions of triggers, A, B, and C, for starting and stopping trace.
Target PC Address	2 Bytes	Select this option to save 5-30% trace-buffer space if your PC addresses never exceed 16-bits (64K). This results in less intrusiveness, or more instructions traced, depending on the trace mode you use. This feature is not supported for the Expert trace mode.
	3 Bytes	Select this default and recommended option if the PC address length in your program exceeds 16-bits (64K). This feature is not supported for the Expert trace mode.
Trace data values	Read Data	Traces the values of data operands being read from the memory. This feature is not supported for the Profile-Only and Expert trace modes.

Table continues on the next page...

Table 4-24. Trace and Profile Options for ColdFire V1 (continued)

Group	Options	Descriptions
	Write Data	Traces the values of data operands being written to the memory. This feature is not supported for the Profile-Only and Expert trace modes.

The following table describes the various Trace and Profile options.

Table 4-25. Trace and Profile Options for Kinetis

Group	Option	Description
ETB		Embedded Trace Buffer where collected trace data is stored.
JTrace		Enables trace collection by using the Segger/J-Trace probe.
	TPIU	Trace Port Interface Unit - Collects ETM and ITM trace into the internal probe buffer of size 4MB. TPIU is a block on the processor that manages the output of trace.
	SWO	Serial Wire Output - Single pin serial output that collects only ITM trace into the buffer of size 4MB. SWO uses the Serial Wire Debug (SWD) debug connection. If selected, the Debug Port Interface should be set as SWD.
	Core clock	ARM core clock in Mhz needed for the serial connection setup. The core clock can change due to multiple settings. For example, when started, a K60 processor rated at 100Mh works at 25Mhz, which is the default value. You can change the core clock value according to the requirements.
Low Power Profiling		Allows monitoring of low power Wait states. This state lets peripherals to function, while allowing CPU to go to sleep reducing power.
Continuous Trace Collection		Allows you to collect continuous trace data when checked. That is, it stops the target in the background to read the trace every time the FIFO is almost full.
ETM		Enables/disables trace output from the Embedded Trace Macrocell (ETM) block. It controls the ETM port selection bit from ETM's control register.
ITM		Enables/disables trace output from the Instrumentation Trace Macrocell (ITM) block.
	Collect Instrumentation trace	Collects instrumentation trace.

Table continues on the next page...

Table 4-25. Trace and Profile Options for Kinetis (continued)

Group	Option	Description
	Collect Profiling Counters	Enables/disables the following profiling counters at once: • Cycle Count Event Generation • Exception Trace • Exception Overhead Count • CPI Count • Sleep Overhead Count • LSU Count • Folded Instruction Count

4.2 Debugging Bareboard Software

This topic applies to debugging software on 10.x systems, that is, for hardware that is not running an operating system.

The topics are:

- Displaying Register Contents
- Exporting Registers
- Importing Registers
- Changing Register Data Display Format
- Offline Registers View
- Using Register Details Window
- Viewing and Modifying Cache Contents
- Setting Stack Crawl Depth
- Changing Program Counter Value
- Viewing Memory
- Hard Resetting

4.2.1 Displaying Register Contents

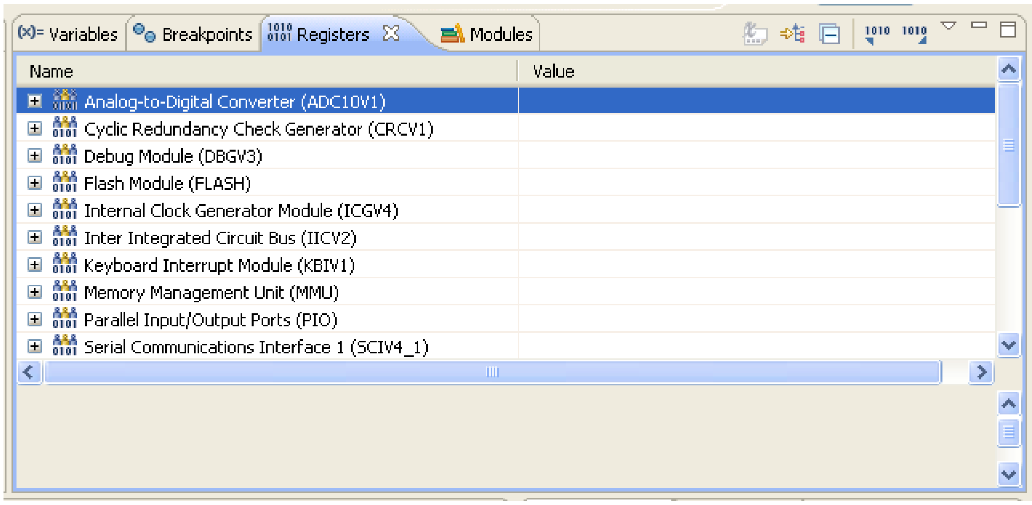
Use the **Registers** view to display and modify the contents of the registers of the processor on your target board. To display this view from the Debug perspective, Select **Window > Show View > Registers** , and the Registers view appears.

The **Registers** view displays categories of registers in a tree format. To display the contents of a particular category of registers, expand the tree element of the register category of interest.

The following figure shows the **Registers** view with the General Purpose Registers tree element expanded.

Tip

You can also view and update registers by issuing the reg, change, and display commands in the CodeWarrior Debugger Shell view.



Name	Value
Analog-to-Digital Converter (ADC10V1)	
Cyclic Redundancy Check Generator (CRCV1)	
Debug Module (DBGV3)	
Flash Module (FLASH)	
Internal Clock Generator Module (ICGV4)	
Inter Integrated Circuit Bus (IICV2)	
Keyboard Interrupt Module (KBIV1)	
Memory Management Unit (MMU)	
Parallel Input/Output Ports (PIO)	
Serial Communications Interface 1 (SCIV4_1)	

Figure 4-24. Registers View

4.2.1.1 Adding Register Group

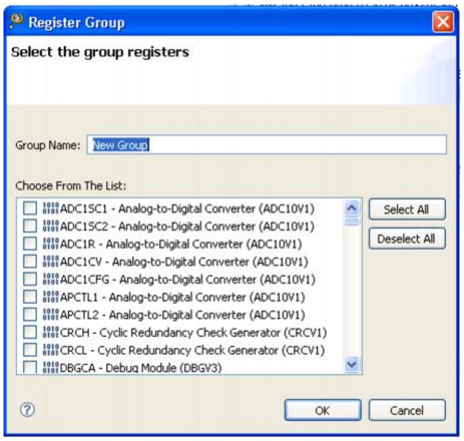
By default, the Registers view lists the related register groups in a tree structure. You can add a custom group of registers to the default tree structure. To add a new register group:

1. Right-click in the Registers view.

   A context menu appears.

2. Select Add Register Group.

   The Register Group dialog box appears.



Register Group

Select the group registers

Group Name: New Group

Choose From The List:

- □ ADC1SC1 - Analog-to-Digital Converter (ADC10V1)
- □ ADC1SC2 - Analog-to-Digital Converter (ADC10V1)
- □ ADC1R - Analog-to-Digital Converter (ADC10V1)
- □ ADC1CV - Analog-to-Digital Converter (ADC10V1)
- □ ADC1CFG - Analog-to-Digital Converter (ADC10V1)
- □ APCTL1 - Analog-to-Digital Converter (ADC10V1)
- □ APCTL2 - Analog-to-Digital Converter (ADC10V1)
- □ CRCH - Cyclic Redundancy Check Generator (CRCV1)
- □ CRCL - Cyclic Redundancy Check Generator (CRCV1)
- □ DBGCA - Debug Module (DBGV3)

Figure 4-25. Register Group Dialog Box

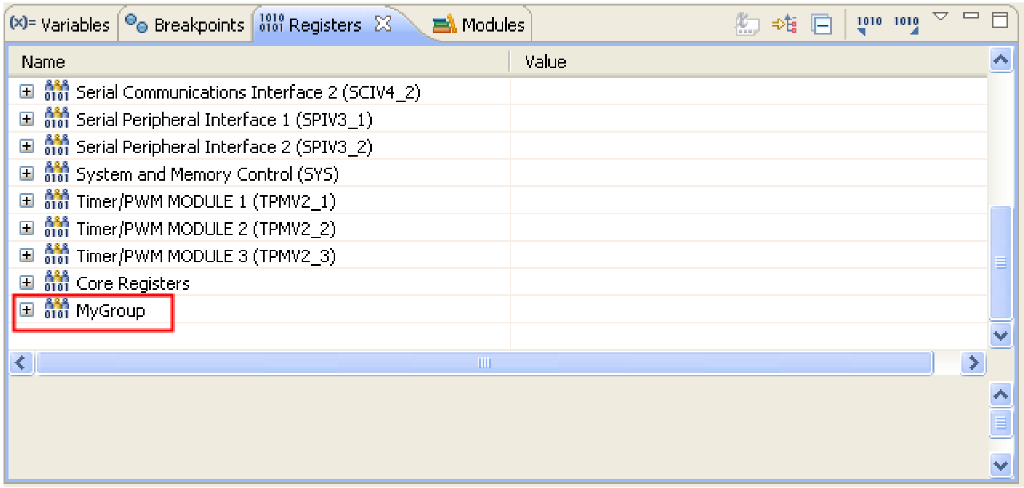
3. Enter in the **Group Name** text box a descriptive name for the new group. For example, *MyGroup*.
4. Check the checkbox adjacent to each register you want to add in the new group.

Tip

Click Select All to check all of the checkboxes. Click Deselect All to clear all the checkboxes.

5. Click OK.

   The Register Group dialog box closes. The new group name appears in the Registers view.



Name	Value
Serial Communications Interface 2 (SCIV4_2)	
Serial Peripheral Interface 1 (SPIV3_1)	
Serial Peripheral Interface 2 (SPIV3_2)	
System and Memory Control (SYS)	
Timer/PWM MODULE 1 (TPMV2_1)	
Timer/PWM MODULE 2 (TPMV2_2)	
Timer/PWM MODULE 3 (TPMV2_3)	
Core Registers	
MyGroup	

Figure 4-26. New Register Group

4.2.1.2 Editing Register Group

In the Registers view, you can edit both the default register groups and the groups that you add. To do so:

1. In the Register view, right-click on the name of the register group you want to edit. For example, right-click on *MyGroup*.

   A context menu appears.

2. Select Edit Register Group.

   The Register Group dialog box appears.

3. If required, enter a new name for the group in the **Group Name** text box.
4. Check the checkbox adjacent to each register you want to add in the group.

   Tip

   Click Select All to check all of the checkboxes. Click Deselect All to clear all the checkboxes.

5. Click OK.

   The Register Group dialog box closes. The new group name appears in the Registers view.

4.2.1.3 Removing Register Group

In the Registers view, you can remove register groups. To remove a register group:

1. In the Registers view, right-click on the register group you want to remove.

   A context menu appears.

2. Select Remove Register Group.

   The selected register group is removed from the Registers view.

4.2.1.3.1 Changing Register's Bit Value

To change a bit value in a register, first switch the IDE to the Debug perspective, start a debugging session and perform these steps.

1. Open the Registers view by selecting Window > Show View > Registers.
2. Expand the register group that contains the register with the bit value that you want to change.

3. Click on the register's current bit value in the view's **Value** column.

   The value appears editable.

4. Type in the new value.
5. Press the **Enter** key.

   The debugger updates the bit value. The bit value in the **Value** column changes to reflect your modification.

NOTE

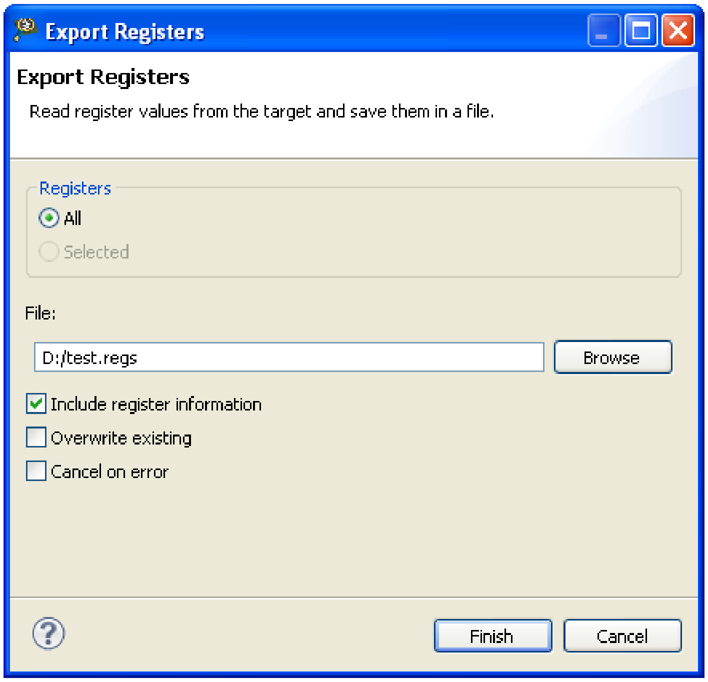
Modified values are highlighted in yellow.

4.2.2 Exporting Registers

To export register data to a file:

1. Open the Registers view.
2. Click the Export registers button in the Registers view toolbar.

   The Export Registers dialog box appears.



Export Registers

Read register values from the target and save them in a file.

Registers

● All

○ Selected

File:

D:/test.regs

☑ Include register information

□ Overwrite existing

□ Cancel on error

Figure 4-27. Export Registers Dialog Box

- Registers group - Controls the scope of export operation. Selecting the All option exports all registers in the Register view. Selecting the Selected option exports selected registers. If a register group is selected in the Register view then the entire register tree, starting at the selected node, is exported.

NOTE

The Selected option is disabled if no register is selected in the Registers view.

- File text box - Specifies the name of the file to store the exported register information.
- Include register information check box - Check this check box to export the location information for registers.
- Overwrite existing check box - Check this check box to overwrite an existing file.
- Cancel on error check box - Check this check box to stop the export operation upon encountering any error.

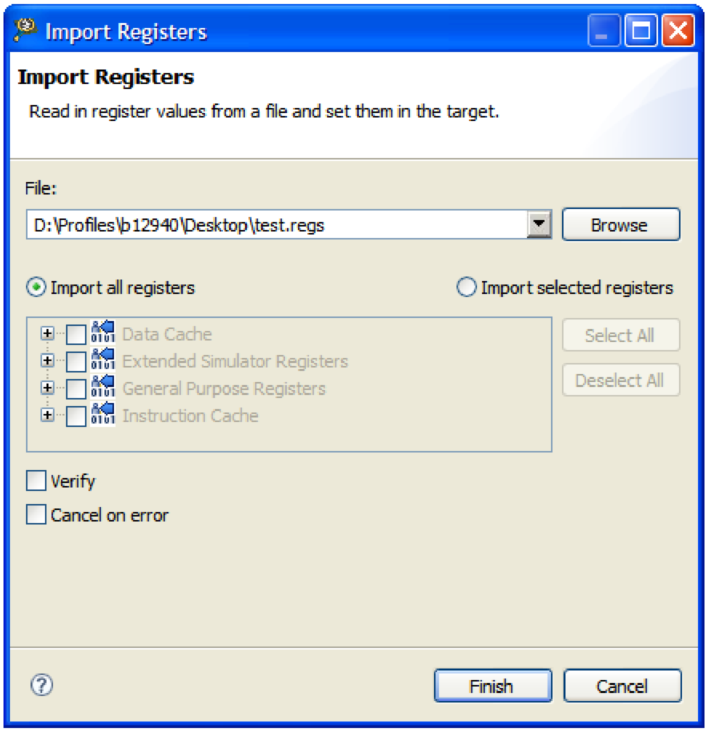
3. Click Finish.

4.2.3 Importing Registers

To import register data from a file:

1. Open the Registers view.
2. Click the Import registers button in the Registers view toolbar.

   The Import Registers dialog box appears.



Import Registers

Import Registers

Read in register values from a file and set them in the target.

File:

D:\Profiles\b12940\Desktop\test.regs

● Import all registers ○ Import selected registers

□ Data Cache
□ Extended Simulator Registers
□ General Purpose Registers
□ Instruction Cache

□ Verify

□ Cancel on error

Figure 4-28. Import Registers Dialog Box

- File drop-down list - Specifies the name of the register data file to import register information.
- Import all registers - Selecting this option allows you to import all registers from the register data file.
- Import selected registers - Selecting this option allows you to select registers you want to import.
- Verify check box - When checked, a register write to the target is followed by a read and a comparison against the written value. This ensures that the import operation on the register is successful.
- Cancel on error check box - Check this check box to stop the import operation upon encountering any error.

3. Click Finish.

4.2.4 Changing Register Data Display Format

You can change the format in which the debugger displays the contents of registers. For example, you can specify that a register's contents be displayed in hexadecimal, rather than binary. The debugger provides these data formats:

- Default
- Decimal
- Hexadecimal

- Octal
- Binary
- Fractional

To change register display format:

1. Open the **Registers** view.
2. Expand the hierarchical list to reveal the register for which you want to change the display format.
3. Select the register value that you want to view in a different format.

   The value highlights.

4. Right-click to display the pop-up menu and and choose **Format > [** *dataformat from the context menu that appears]*, where *dataformat* is the data format in which you want to view the register value.

   The register value changes format.

4.2.5 Offline Registers View

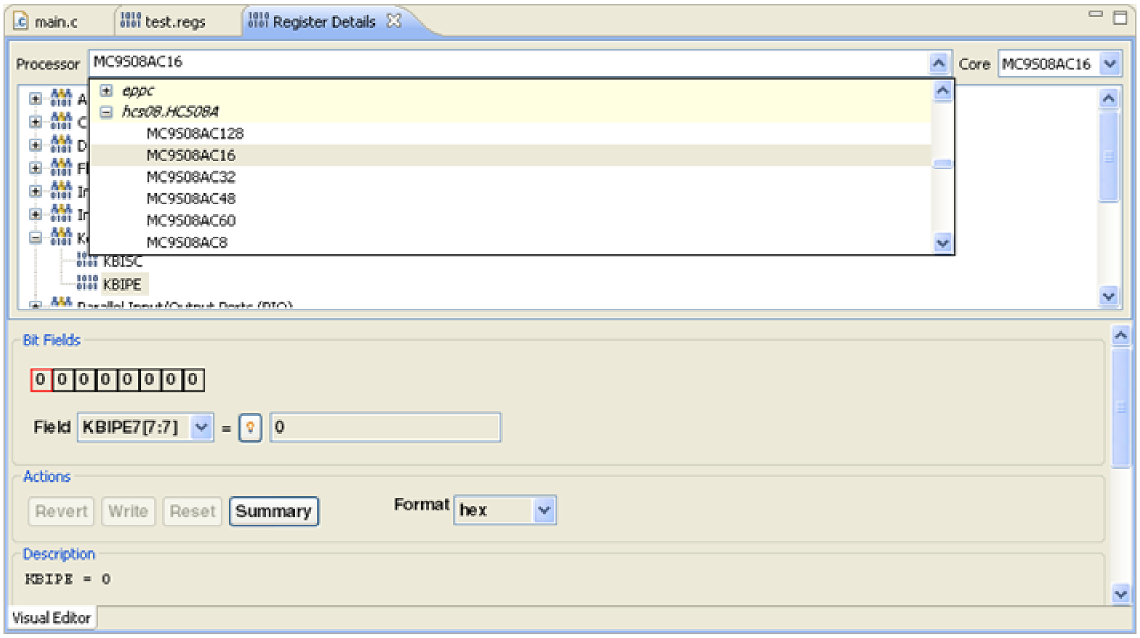
The Offline Registers View feature allows you to browse registers information from debugger database offline (without a debug session). Register details are presented in the same way as in registers view.

It also provides an editor for "regs" files (files exported from registers view). You can browse the registers from file offline. You can update register values in visual and text mode for later importing data through debugger.

To open offline registers view:

1. Click Offline Registers View button on the toolbar.

   The Register Details page appears.



Processor MC9S08AC16

Core MC9S08AC16

Field KBIPE7[7:7] = 0

Format hex

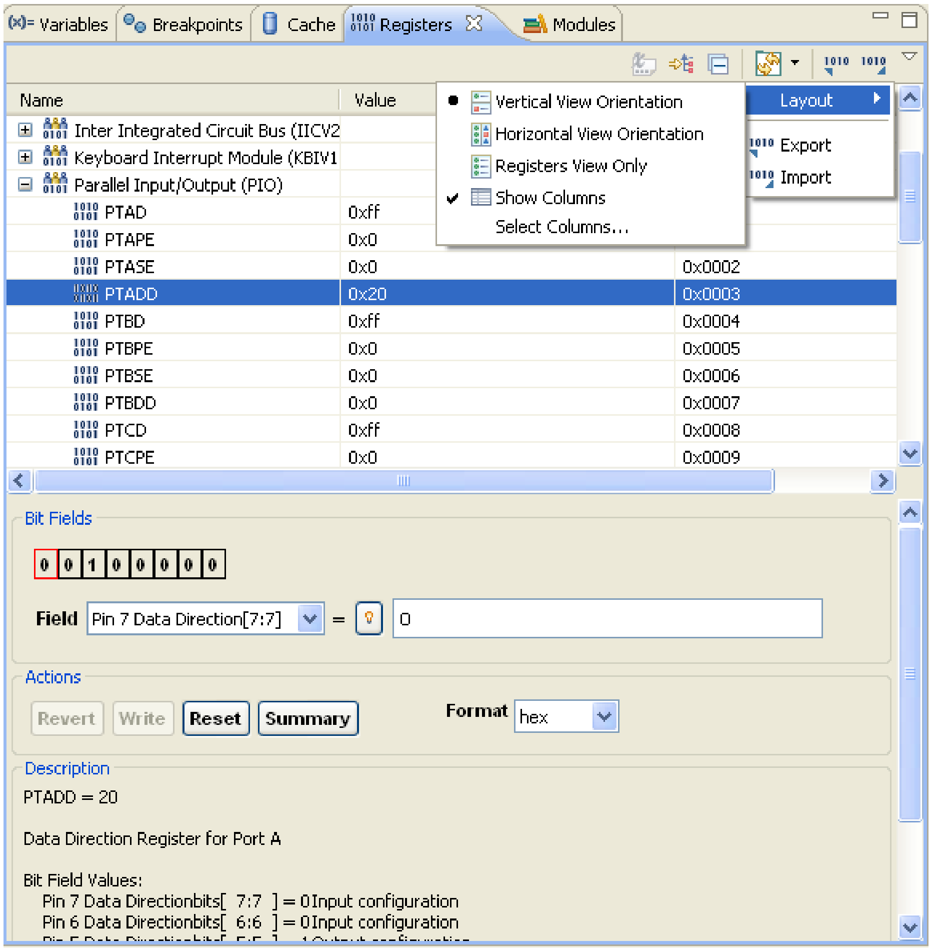
KBIPE = 0

Figure 4-29. Offline Registers View - Register Details

2. Select a chip name from the Processor drop-down list.
3. Click the register from the list to see details like bit fields, actions and description.

4.2.6 Using Register Details Window

The default state of the Registers view is to provide details on the processor's registers.



Name	Value	
Inter Integrated Circuit Bus (IICV2		
Keyboard Interrupt Module (KBIV1		
Parallel Input/Output (PIO)		
PTAD	0xff	
PTAPE	0x0	
PTASE	0x0	0x0002
PTADD	0x20	0x0003
PTBD	0xff	0x0004
PTBPE	0x0	0x0005
PTBSE	0x0	0x0006
PTBDD	0x0	0x0007
PTCD	0xff	0x0008
PTCPE	0x0	0x0009

0 0 1 0 0 0 0 0

Field: Pin 7 Data Direction[7:7] = 0

Format: hex

PTADD = 20

Data Direction Register for Port A

Bit Field Values:
Pin 7 Data Directionbits[7:7] = 0Input configuration
Pin 6 Data Directionbits[6:6] = 0Input configuration
Pin 5 Data Directionbits[5:5] = 1Output configuration

Figure 4-30. Register View - Detailed Information

The Registers view displays several types of register details:

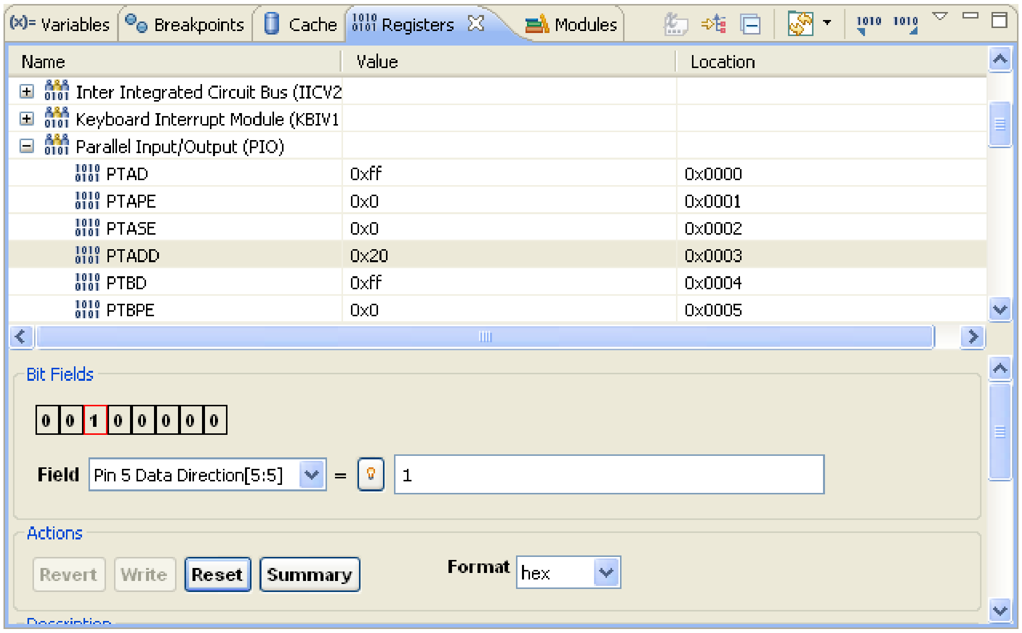
- Bit Fields
- Description
- Actions

NOTE

To display the register details, first you have to select a register, then expand the view by clicking and dragging the areas at the bottom of the Registers view to reveal the Bit Field, Description, and Actions portions of the view.

4.2.6.1 Bit Fields

The Bit Fields group of the Registers view shows a graphical representation of the selected register's bit values. This graphical representation shows how the register organizes bits. You can use this representation to select and change the register's bit values. Hover the cursor over each part of the graphical representation in order to see additional information.



Name	Value	Location
Inter Integrated Circuit Bus (IICV2		
Keyboard Interrupt Module (KBIV1		
Parallel Input/Output (PIO)		
PTAD	0xff	0x0000
PTAPE	0x0	0x0001
PTASE	0x0	0x0002
PTADD	0x20	0x0003
PTBD	0xff	0x0004
PTBPE	0x0	0x0005

0 0 1 0 0 0 0 0

Field: Pin 5 Data Direction[5:5] = 1

Format: hex

Figure 4-31. Register Details - Bit Fields Group

Tip

> You can also view register details by issuing the `reg` command in the Debugger Shell view.

A bit field is either a single bit or a collection of bits within a register. Each bit field has a mnemonic name that identifies it. You can use the Field list box to view and select a particular bit field of the selected register. The list box shows the mnemonic name and bit-value range of each bit field. In the Bit Fields graphical representation, a box surrounds each bit field. A red box surrounds the bit field shown in the Field list box.

After you use the Field list box to select a particular bit field, you see its current value in the = text box. If you change the value shown in the text box, the Registers view shows the new bit-field value.

The minimum resolution of bit-field descriptions is 2-bits. Consequently, register details are not available for single-bit overflow registers.

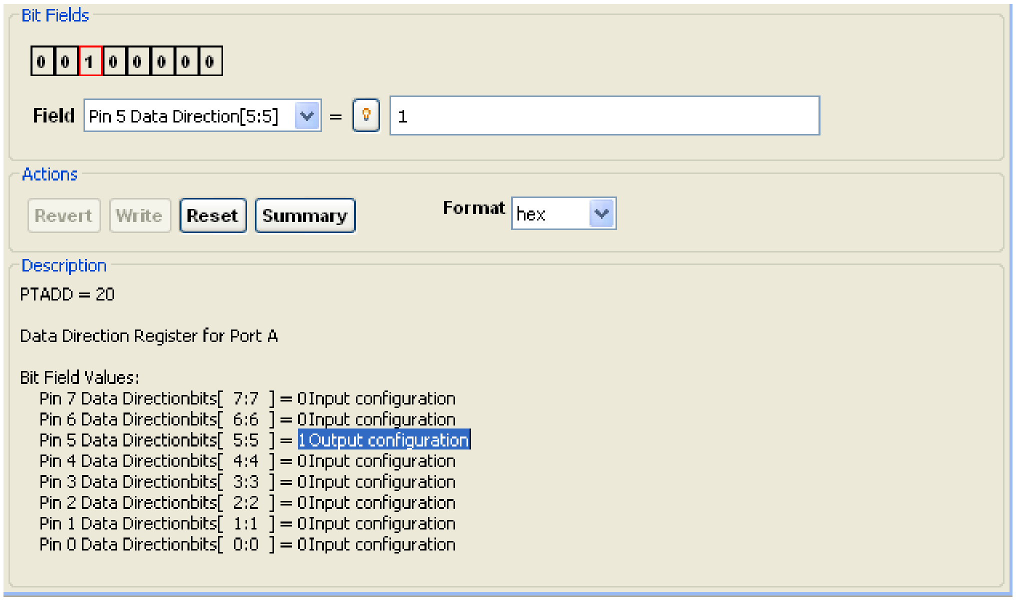
The maximum resolution of bit-field descriptions is 32-bits.

4.2.6.1.1 Changing Bit Field

To change a bit field in a register, you must first start a debugging session and then open the Registers view.

To change a bit field, perform these steps.

1. In the Registers view, view register details.
2. Expand the register group that contains the bit field you want to change.
3. Register details appear in the Registers view.



```
PTADD = 20

Data Direction Register for Port A

Bit Field Values:
   Pin 7 Data Directionbits[  7:7  ] = 0Input configuration
   Pin 6 Data Directionbits[  6:6  ] = 0Input configuration
   Pin 5 Data Directionbits[  5:5  ] = 1Output configuration
   Pin 4 Data Directionbits[  4:4  ] = 0Input configuration
   Pin 3 Data Directionbits[  3:3  ] = 0Input configuration
   Pin 2 Data Directionbits[  2:2  ] = 0Input configuration
   Pin 1 Data Directionbits[  1:1  ] = 0Input configuration
   Pin 0 Data Directionbits[  0:0  ] = 0Input configuration
```

Figure 4-32. Registers View - Register Details

4. From the expanded register group above the register details, select the name of the register that contains the bit field that you want to change.

   The Bit Fields group displays a graphical representation of the selected bit field. The **Description** group displays explanatory information about the selected bit field and parent register.

5. In the **Bit Fields** group, click the bit field that you want to change. Alternatively, use the Field list box to specify the bit field that you want to change.
6. In the = text box, type the new value that you want to assign to the bit field.
7. In the Action group, click the Write button.

NOTE

The **Revert** and **Write** buttons appear enabled if the value in the **=** field is changed or you reset the values.

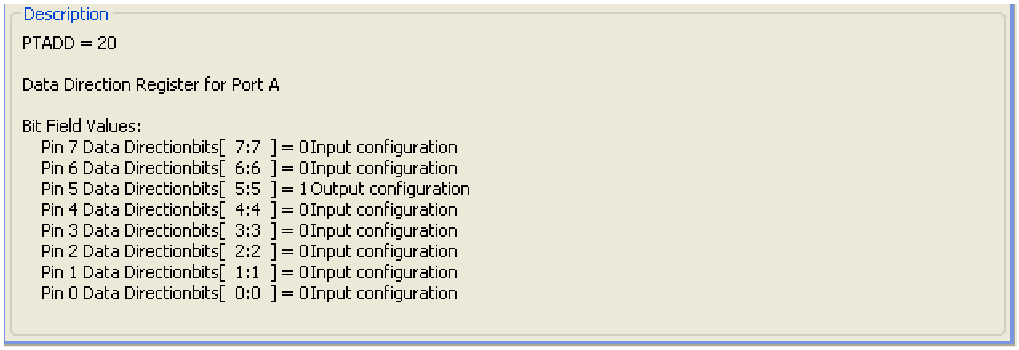
The debugger updates the bit-field value. The bit values in the **Value** column and the **Bit Fields** group change to reflect your modification.

NOTE

Click the Reset button to discard your changes and restore the original bit-field value. Click the **Revert** button to revert to the last changes made.

4.2.6.2 Description

The Description group of the Registers view shows explanatory information for the selected register.



```
Description
PTADD = 20

Data Direction Register for Port A

Bit Field Values:
   Pin 7 Data Directionbits[  7:7  ] = 0Input configuration
   Pin 6 Data Directionbits[  6:6  ] = 0Input configuration
   Pin 5 Data Directionbits[  5:5  ] = 1Output configuration
   Pin 4 Data Directionbits[  4:4  ] = 0Input configuration
   Pin 3 Data Directionbits[  3:3  ] = 0Input configuration
   Pin 2 Data Directionbits[  2:2  ] = 0Input configuration
   Pin 1 Data Directionbits[  1:1  ] = 0Input configuration
   Pin 0 Data Directionbits[  0:0  ] = 0Input configuration
```

Figure 4-33. Register View - Description Group

The registers information covers:

- Name
- Current Value
- Description
- Bit field explanations and values

Some registers have multiple modes (meaning that the register's bits can have multiple meanings, depending on the current mode). If the register you examine has multiple modes, you must select the appropriate mode.

4.2.6.3 Actions

Use the Actions group of the Registers view to perform various operations on the selected register's bit-field values.

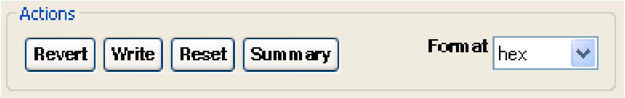


Figure 4-34. Register View - Actions Group

The following table lists each item in the **Actions** group and explains the purpose of each.

Table 4-26. Actions Group Items

Item	Description
Revert	Discard your changes to the current bit-field value and restore the last change you made. The debugger disables this button if you have not made any changes to the bit-field value.
Write	Save your changes to the current bit-field value and write those changes into the register's bit field. The debugger disables this button after writing the new bit-field value, or if you have not made any changes to that value.
Reset	Change each bit of the bit-field value to its register-reset value. The register takes on this value after a target-device reset occurs. To confirm the bit-field change, click Write. To cancel the change, click Reset **.**
Summary	Display **Description** group content in a pop-up window. Press the Esc key to close the pop-up window.
Format	Specify the data format of the displayed bit-field values.

4.2.6.4 Register Details Context Menu

To display the **Register Details** context menu, right-click on a bit-field value in the Registers view. This menu's commands duplicate capabilities available in the Register Details view.

The following table lists each command in the Registers view and explains the purpose of each.

Table 4-27. Register Details Context Menu

Menu Command	Description
Select All	Selects the entire contents of the current bit-field value
Copy Registers	Copies to selected bit fields content to the system clipboard
Enable	Lets the debugger to access the selected bit fields
Disable	Prevents the debugger from accessing the selected bit fields
Format	Formats the register value as decimal, hexadecimal, octal, binary, or fractional.
Cast to Type	Opens a dialog box that you can use to cast the selected bit field values to a different data type
Restore Original Type	Reverts the selected bit-field values to their default data types
View Memory	Opens the Memory view and provides memory details of the selected register.
Find	Opens a dialog box that you can use to select a particular register or bit field

Table continues on the next page...

Table 4-27. Register Details Context Menu (continued)

Menu Command	Description
Change Value	Open the Set Value dialog to change the selected registers value.
Show Details As	Helps you view details in the default viewer or in the registers details pane. In case default viewer is selected, teh bit description is not displayed as in case of the registers details pane.
Add Register Group	Opens a dialog box that you can use to create a new collection of registers to display in the Registers view
Restores Default Register Group	Resets the custom groups of registers created using the Add Register Group option, and restores the default groups provided by the debugger as they were when CodeWarrior was installed. Note that if you select this option, all custom groupings of registers done by you are lost.
Add Watchpoint (C/C++)	Helps you add watchpoint to stop the execution of an application whenever the value of a given expression changes, without specifying where it might occur.
Edit Register Group	Opens a dialog box that you can use to modify the collection of registers that the Registers view displays for the selected register group
Remove Register Group	Deletes the selected register group from the Registers view
Profile As	Helps you save your current profile.
Watch	Opens the Expressions view.
Resource Configurations	Helps you exclude or include resource configurations from the build.
Add Visualization	Enables you to specify the attributes for register read operations in the Add Data Visualization dialog.

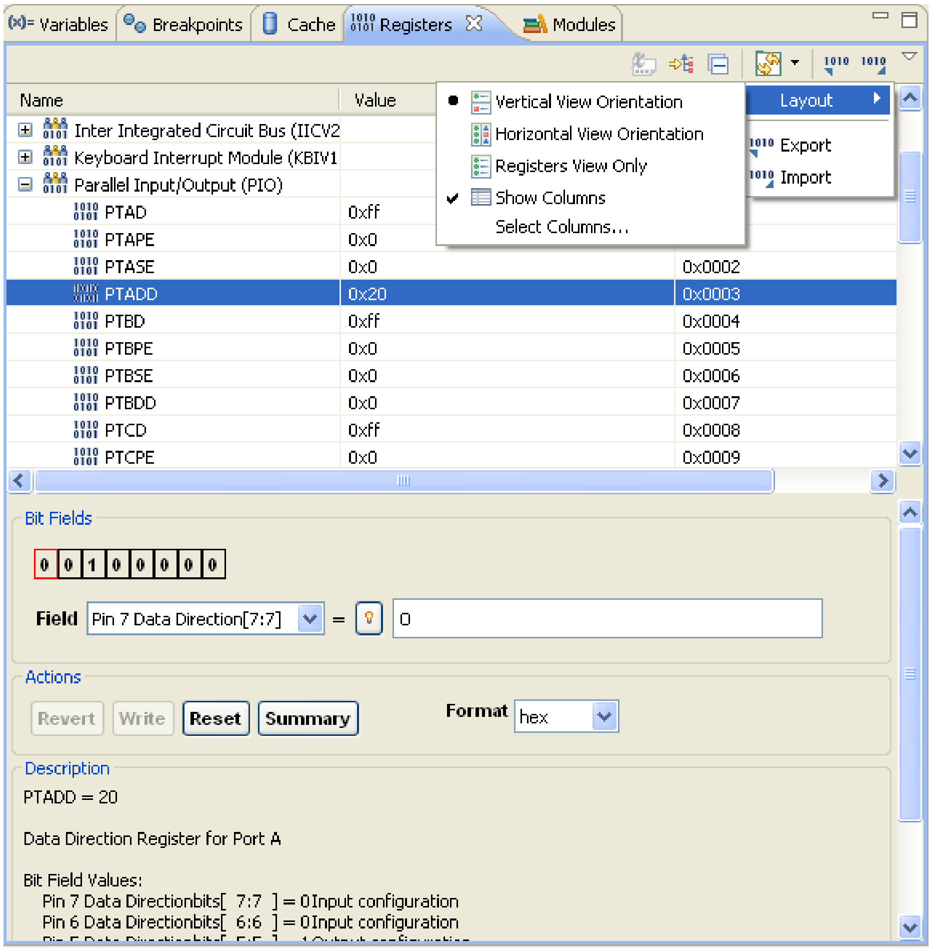
4.2.6.5 Viewing Register Details

To open the **Registers** view, you must first start a debugging session.

To see the registers and their descriptions, perform these steps.

1. In the **Debug** perspective, click the **Registers** tab.

   The **Registers** view appears.



Name	Value	
⊞ Inter Integrated Circuit Bus (IICV2		
⊞ Keyboard Interrupt Module (KBIV1		
⊟ Parallel Input/Output (PIO)		
PTAD	0xff	
PTAPE	0x0	
PTASE	0x0	0x0002
PTADD	0x20	0x0003
PTBD	0xff	0x0004
PTBPE	0x0	0x0005
PTBSE	0x0	0x0006
PTBDD	0x0	0x0007
PTCD	0xff	0x0008
PTCPE	0x0	0x0009

0 0 1 0 0 0 0 0

Field Pin 7 Data Direction[7:7] = 0

Format hex

PTADD = 20

Data Direction Register for Port A

Bit Field Values:
Pin 7 Data Directionbits[7:7] = 0Input configuration
Pin 6 Data Directionbits[6:6] = 0Input configuration

Figure 4-35. Registers View, Register Details

2. Click the toolbar's menu button (the inverted triangle highlighted in the above figure).
3. Select **Layout > Vertical View Orientation** or **Layout > Horizontal View Orientation** to see the register details.

NOTE

Selecting **Layout > Registers View Only** hides the register details.

4. Expand a register group to see individual registers.
5. Select a specific register by clicking on it.

   The debugger enables the appropriate buttons in the Actions group of the **Registers** view.

NOTE

Use the **Format** list box to specify the format of data that appears in the **Registers** view.

6. Use the **Register** view to examine register details.

For example, examine register details in these ways:

- Use the **Bit Fields** group to see a graphical representation of the selected register's bit fields. You can use this graphical representation to select specific bits or bit fields.
- Use the **Description** group to see an explanation of the selected register, bit field, or bit value.

Tip

To enlarge the **Registers** view, click **Maximize** of the view's toolbar. After you finish looking at the register details, click **Restore** of the view's toolbar to return the view to its previous size. Alternatively, right-click the **Registers** tab and select **Detached** . The **Registers** view becomes a floating window that you can resize. After you finish looking at the register details, right-click the **Registers** tab of the floating window and select **Detached** again. You can rearrange the re-attached view by dragging its tab to a different collection of view tabs.

4.2.7 Viewing and Modifying Cache Contents

The CodeWarrior debugger lets you view and modify the instruction cache and data cache of the target system during a debug session.

- Cache Viewer
- Cache Viewer Toolbar Menu
- Components of Cache Viewer
- Using the Debugger Shell to View Caches
- Supported Processor Cache Features

4.2.7.1 Cache Viewer

Use the cache viewer to examine L1 cache (such as instruction cache or data cache). Also, you can use the viewer to display L2 and L3 cache for targets that support it.

4.2.7.1.1 Opening the Cache Viewer

To open the cache viewer:

1. Start a debugging session.

2. From the main menu bar, select **Window > Show View > Other** .

   The Show View dialog box appears as shown in the following figure.

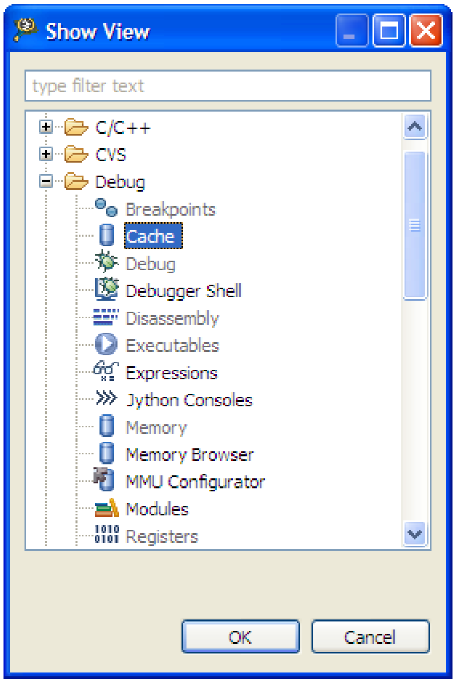


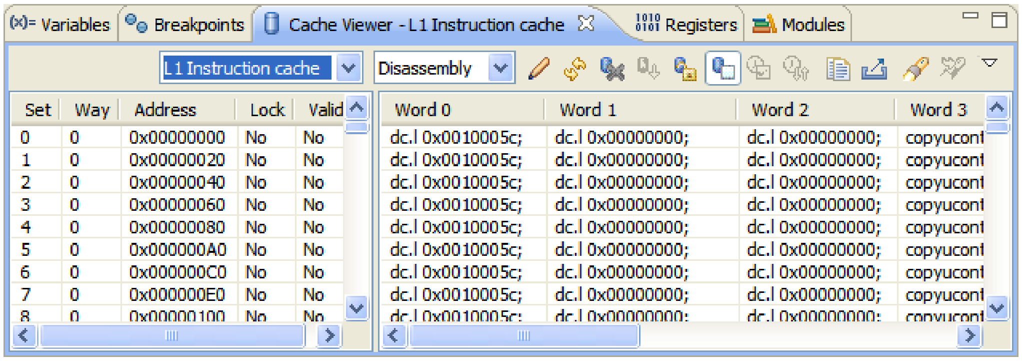
Figure 4-36. Show View Dialog Box

3. Expand the Debug group.
4. Select Cache.
5. Click OK.

   The Cache Viewer view appears as shown in the following figure.

Tip

You can use the type filter text box as a shortcut to specify Cache Viewer. Start typing cache viewer into the text box. The Show View dialog box shortens the list of views to those whose names match the characters you type. The list continues to shorten as you type each additional character. When the list shows just the Cache Viewer view, select it and click OK to open that view. You can click the Clear button to empty the text box and restore the full list of views.



Set	Way	Address	Lock	Valid
0	0	0x00000000	No	No
1	0	0x00000020	No	No
2	0	0x00000040	No	No
3	0	0x00000060	No	No
4	0	0x00000080	No	No
5	0	0x000000A0	No	No
6	0	0x000000C0	No	No
7	0	0x000000E0	No	No
8	0	0x00000100	No	No

Word 0	Word 1	Word 2	Word 3
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont
dc.l 0x0010005c;	dc.l 0x00000000;	dc.l 0x00000000;	copyucont

Figure 4-37. Cache Viewer

6. Use the Choose a Cache list box to specify the cache that you want to examine.

NOTE

If the Choose a Cache list box is grayed out, the current target does not support viewing cache.

4.2.7.2 Cache Viewer Toolbar Menu

Click the Cache Viewer toolbar menu button (the inverted triangle) to modify the data display. The cache viewer toolbar buttons are alternate ways to implement these control actions.

The following tab;e describes the Cache Viewer toolbar menu options.

NOTE

Certain toolbar buttons are unavailable (grayed out) if the target hardware does not support their corresponding functions, or if a specific operation can be performed in assembly language and is not supported by the cache viewer.

Table 4-28. Cache Viewer Toolbar Menu Options

Option	Description
Write	Commit content changes from the cache viewer to the cache registers on the target hardware (if the target hardware supports doing so)
Refresh	Read data from the target hardware and update the cache viewer display
Invalidate	Invalidate the entire contents of the cache
Flush	Flush the entire contents of the cache. This option commits uncommitted data to the next level of the memory hierarchy, then invalidates the data within the cache.
Lock	Lock the cache and prevent the debugger from fetching new lines or discarding current valid lines

Table continues on the next page...

Table 4-28. Cache Viewer Toolbar Menu Options (continued)

Option	Description
Enable	Turn on the cache
Disable LRU	Remove the Least Recently Used (LRU) attribute from the existing display for each cache line
Inverse LRU	Display the inverse of the Least Recently Used attribute for each cache line
Copy Cache	Copy the cache contents to the system clipboard
Export Cache	Export the cache contents to a file
Search	Find an occurrence of a string in the cache lines
Search Again	Find the next occurrence of a string in the cache lines
View Memory	View the corresponding memory for the selected cache lines
Lock Line	Lock the selected cache lines
Invalidate Line	Invalidate the selected cache lines
Flush Line	Flush the entire contents of the selected cache lines
Lock Way	Lock the cache ways specified with the Lock Ways menu command. Locking a cache way means that the data contained in that way must not change. If the cache needs to discard a line, it will not discard locked lines (such as lines explicitly locked, or lines belonging to locked ways).
Unlock Ways	Unlock the cache ways specified with the Lock Ways menu option
Lock Ways	Specify the cache ways on which the Lock Way and Unlock Way menu commands operate.

4.2.7.3 Components of Cache Viewer

Below the toolbar, there are two panes in the window, separated by another vertical divider line. The pane to the left of the divider line displays the attributes for each displayed cache line. The pane to the right of the divider line displays the actual contents of each displayed cache line. You can modify information in this pane and click the Write button to apply those changes to the cache on the target board.

Above the cache line display panes are the Refresh and Write buttons and the View As dropdown menu. Click the Refresh button to clear the entire contents of the cache, re-read status information from the target hardware, and update the cache lines display panes. Click the Write button to commit cache content changes from this window to the cache memory on the target hardware (if the target hardware supports doing so). Select Raw Data, or Disassembly from the View As dropdown menu to change the way the IDE displays the data in the cache line contents pane on the right side of the window.

You can perform all cache operations from assembly code in your programs. For details about assembly code, refer to the core documentation for the target processor.You can also perform cache operations by clicking the Menu button, shown as an inverted triangle, that opens the view pull-down menu that contain actions for the Cache Viewer.

4.2.7.4 Using the Debugger Shell to View Caches

Another way to manipulate the processor's caches is by using the Debugger Shell.

To display the Debugger Shell:

1. Start a debugging session.
2. Select Window > Show View > Other.

   The Show View dialog box appears.

3. Expand the Debug group.
4. Select Debugger Shell.
5. Click the OK button.

   The Debugger Shell view appears.

To display a list of the commands supported by the Debugger Shell, enter this at the command prompt:

```
help -tree
```

For more information about the Debugger Shell support of cache commands, enter these commands at the command prompt:

```
help cmdwin::ca
```

```
help cmdwin::caln
```

The next sections describe these commands in more detail.

4.2.7.4.1 Debugger Shell Global Cache Commands

The `cmdwin::ca` cache commands manage global cache operations. That is, they affect the operation of the entire cache. For multi-core processors, these commands operate on a specific cache if an optional ID number is provided. If the ID number is absent, the command operates on the cache that was assigned as the default by the last cmdwin::ca::default command.

The following table summarizes these cache commands.

Table 4-29. Debugger Shell Global Cache Commands

Command	Description
cmdwin::ca::default	Set specified cache as default
cmdwin::ca::enable	Enable/disable cache
cmdwin::ca::flush	Flushes cache
cmdwin::ca::inval	Invalidates cache
cmdwin::ca::lock	Lock/Unlock cache
cmdwin::ca::show	Show the architecture of the cache

The basic format of a shell global cache command is:

command [<cache ID>] [on | off]

The optional cache ID number argument selects the cache that the command affects.

The optional on or off argument changes a cache's state.

For example, to display a particular cache's characteristics:

%> cmdwin::ca:show 1

displays the characteristics of the second processor cache.

You use the cmd::ca::default to assign a default cache that becomes the target of global cache commands. For example:

%> cmdwin::ca:default 0

makes the first processor cache the default cache. Subsequent global cache commands that do not specify a cache ID will affect this cache.

Other cache commands require the off or on state argument. When specifying a particular cache, the state argument follows the ID argument. For example:

%> cmdwin::ca:lock 2 on

locks the contents of the third processor cache, while:

%> cmdwin::ca:enable 1 off

disables the second processor cache.

4.2.7.4.2 Debugger Shell Cache Line Commands

The `cmdwin::caln` commands manage cache line operations. They affect memory elements within a designated cache.

The following table summarizes these commands.

Table 4-30. Debugger Shell Cache Line Commands

Command	Description
cmdwin::caln::get	Display cache line
cmdwin::caln::flush	Flush cache line
cmdwin::caln::inval	Invalidate cache line
cmdwin::caln::lock	Locks/unlocks cache line
cmdwin::caln::set	Writes specified data to cache line

The basic format for a shell cache line command is:

command [<cache ID>] <line> [<count>]

The optional cache ID argument specifies the cache that the command affects, otherwise it affects the default cache, as set by the cmdwin::ca::default command.

The required line argument specifies the cache line to affect.

The optional count argument specifies the number of cache lines the command affects. The default is one line. For example:

%> cmdwin::caln:flush 2

flushes line 2 of the default cache.

The cmdwin::caln:set command varies from the other commands in that you must specify data words that fill the cache line. For example:

%> cmdwin::caln:set 2 = 0 1 1 2 3 5 8 13

Sets the contents of cache line two, where the first word has a value of 0, the second word has a value of 1, the third word has a value of 1, the fourth word has a value of 2, and so on.

NOTE

If the command specifies a list of data values that are less than one line's worth of words, then the values are repeated from the beginning of the list to complete the filling the cache line. If too many data words are specified for the cache line to hold, the extra values are discarded.

4.2.7.5 Supported Processor Cache Features

This section lists the cache features and supported status flags.

Table 4-31. P4080 - Supported Cache Operations and Status Flags

Cache	Features	Supported Operations	Supported Status Flags
L1 data cache	• 32 KB size • 64 sets • 8 ways • 16 words / line	• enable/disable cache • lock/unlock cache • invalidate cache • lock/unlock line • invalidate line • read/modify data • flush cache • flush line	• valid • lock • shared • dirty • castout • plru
L1 instruction cache	• 32 KB size • 64 sets • 8 ways • 16 words / line	• enable/disable cache • lock/unlock cache • invalidate cache • lock/unlock line • invalidate line • read/modify data	• valid • lock • plru
L2 cache	• 128 KB size • 256 sets • 8 ways • 16 words / line	• enable/disable cache • lock/unlock cache • invalidate cache • lock/unlock line • invalidate line • read/modify data • flush cache • flush line	• valid • lock • shared • dirty • noncohe-rent • plru
L3 cache	• 2 banks • 512KB/bank • 512 sets • 32 ways • 16 words/line	• enable/disable cache • lock/unlock cache • invalidate cache • lock/unlock line • invalidate line • read/modify data • flush cache • flush line	• valid • locked • modified • plru

4.2.8 Setting Stack Crawl Depth

Select the Maximum stack crawl depth command lets you set the depth of the stack to read and display. Showing all levels of calls when you are examining function calls several levels deep can sometimes make stepping through code more time-consuming. Therefore, you can use this menu option to reduce the depth of calls that the debugger displays.

To set the stack crawl depth, perform these steps.

1. Select Window > Preferences.

   The Preferences dialog box appears.

2. Expand the **C/C++** tree control and select Debug.

   The general settings for C/C++ debugging are displayed on the right-hand side of the Preferences dialog box.

3. Specify the appropriate stack crawl depth, in the Maximum stack crawl depth text box.

NOTE

You can specify any integer from 1 to 100.

4.2.9 Changing Program Counter Value

To change the program-counter value, perform these steps.

1. Start a debugging session.
2. In the **Editor** view, place the cursor on the line that you want the debugger to execute next.
3. Right-click in the **Editor** view.

   A context menu appears.

4. From the context menu, select **Move To Line** .

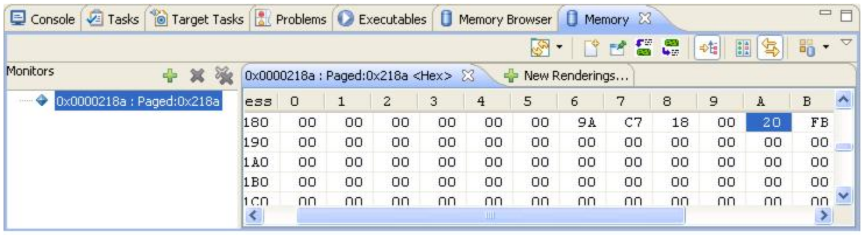
   CodeWarrior IDE modifies the program counter to the specified location. The **Editor** view shows the new location.

4.2.10 Viewing Memory

Use the **Memory** view to examine the active memory rendering of a specified expression or address. To display this view from the Debug perspective, Select **Window > Show View > Memory** , and the **Memory** view appears.

The **Memory** view supports the display of multiple memory spaces.

The following figure shows the **Memory** view with the *Expression:baseaddr <name> tree* active memory rendering tab.



ess	0	1	2	3	4	5	6	7	8	9	A	B
180	00	00	00	00	00	00	9A	C7	18	00	20	FB
190	00	00	00	00	00	00	00	00	00	00	00	00
1A0	00	00	00	00	00	00	00	00	00	00	00	00
1B0	00	00	00	00	00	00	00	00	00	00	00	00
1C0	00	00	00	00	00	00	00	00	00	00	00	00

Figure 4-38. Memory View

4.2.10.1 Adding Memory Monitor

You can add multiple memory monitors to the Memory view. To add a new memory monitor, perform these steps.

1. Start a debugging session.
2. Open the Memory view.
3. Click the plus-sign

   ✚

   icon on the Monitors pane toolbar. Alternatively, right-click in the Monitors pane and select Add Memory Monitor from the context menu.
4. The Monitor Memory dialog box appears.

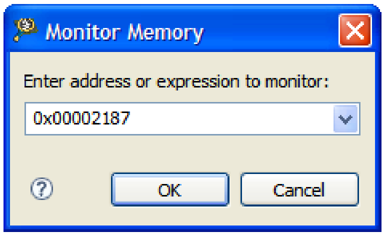
NOTE

> The Enter memory space and address option appears only when the debugger associated with the active debugging context supports memory spaces, and the currently debugged process has multiple memory spaces.

5. Specify options as explained in the following table.

Table 4-32. Monitor Memory Dialog Box Options

Option	Description
Enter address or expression	Enter the expression to monitor in decimal or hexadecimal values. You can use the drop-down list to select a previously specified expression.



Monitor Memory

Enter address or expression to monitor:

0x00002187

Figure 4-39. Monitor Memory Dialog Box

6. Click OK.

   The memory monitor appears in the Memory view.

4.2.10.2 Adding Memory Rendering

You can use the Renderings pane of the Memory view to examine the memory content, starting at any valid address. The information displayed in this page is read only and cannot be used to modify the memory content.

To add a new memory rendering, perform these steps.

1. Start a debugging session.
2. Open the Memory view.
3. In the Monitors pane, select the memory monitor for which you want to add a memory rendering.

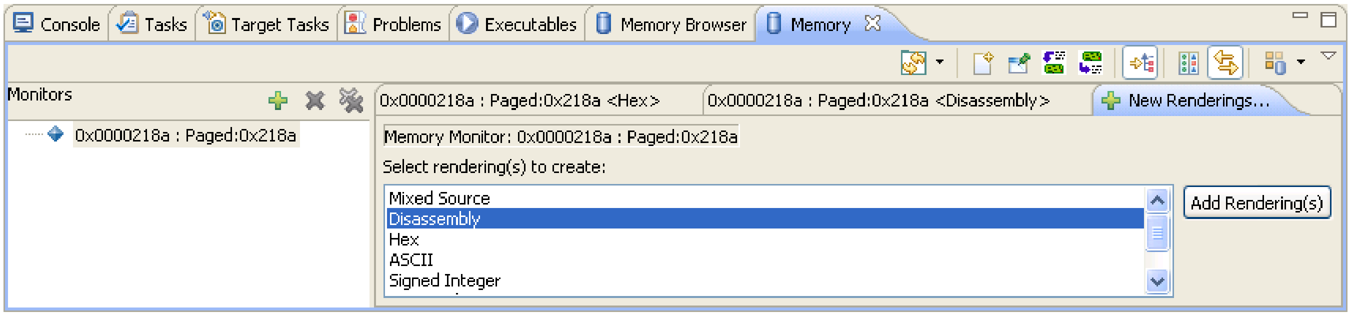
NOTE

To create a memory monitor, right-click a blank area in the Monitors pane and select Add Memory Monitor. Alternatively, click the plus-sign

+

icon in the Monitors pane toolbar.

4. Click the **New Renderings** tab to select renderings.



Memory Monitor: 0x0000218a : Paged:0x218a

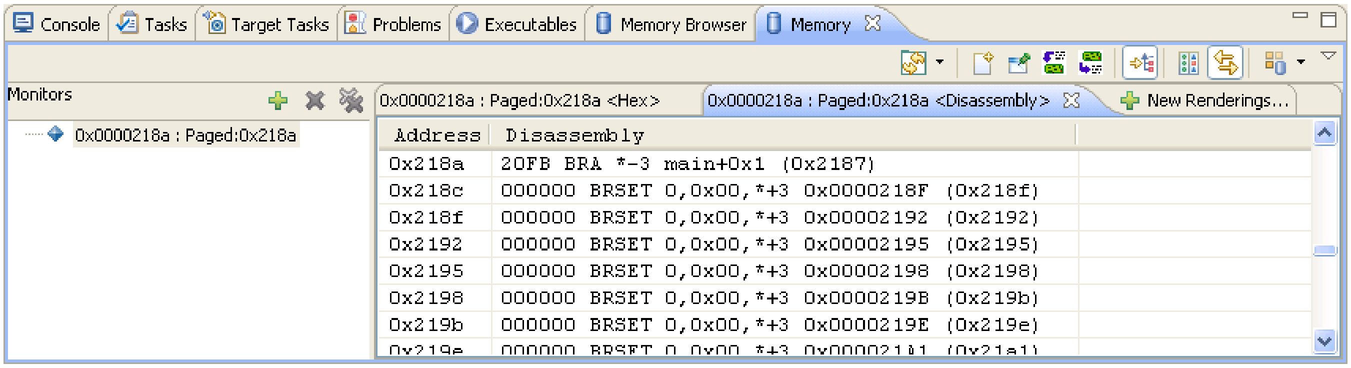
Select rendering(s) to create:

- Mixed Source
- Disassembly
- Hex
- ASCII
- Signed Integer

Figure 4-40. New Renderings View

5. Select a rendering type from the **Select rendering(s) to create** list and click the **Add Rendering(s)** button. Alternatively, right-click in the Renderings pane and select Add Rendering from the context menu. For example, select *Disassembly*.
6. Click **Rendering(s)** .

   The selected memory rendering type appears in the Memory view.


Console
Tasks
Target Tasks
Problems
Executables
Memory Browser
Memory
Monitors
0x0000218a : Paged:0x218a
0x0000218a : Paged:0x218a <Hex>
0x0000218a : Paged:0x218a <Disassembly>
New Renderings...

```
Address   Disassembly
0x218a    20FB BRA *-3 main+0x1 (0x2187)
0x218c    000000 BRSET 0,0x00,*+3 0x0000218F (0x218f)
0x218f    000000 BRSET 0,0x00,*+3 0x00002192 (0x2192)
0x2192    000000 BRSET 0,0x00,*+3 0x00002195 (0x2195)
0x2195    000000 BRSET 0,0x00,*+3 0x00002198 (0x2198)
0x2198    000000 BRSET 0,0x00,*+3 0x0000219B (0x219b)
0x219b    000000 BRSET 0,0x00,*+3 0x0000219E (0x219e)
0x219e    000000 BRSET 0,0x00,*+3 0x000021A1 (0x21a1)
```

Figure 4-41. Added Rendering

4.2.10.3 Removing Memory Rendering

To remove a memory rendering from the Memory view, perform these steps.

1. Open the Memory view.
2. In the Renderings pane, select the tab that corresponds to the memory rendering that you want to remove.
3. Click the cross-sign

   ✖

   icon on the Renderings pane toolbar. Alternatively, right-click on the Renderings pane and select Remove Rendering from the context menu.

   The memory rendering is removed from the **Memory** view.

4.2.10.4 Resetting to Base Address

To reset the memory rendering and display the base address of the rendering, perform these steps.

1. Open the Memory view.
2. In the Renderings pane, select the tab that corresponds to the disassembly rendering that you want to reset to the base address.
3. Right-click in the Renderings pane and select Reset to Base from the context menu.
4. The disassembly rendering scrolls to the line that contains the base address of the displayed rendering.

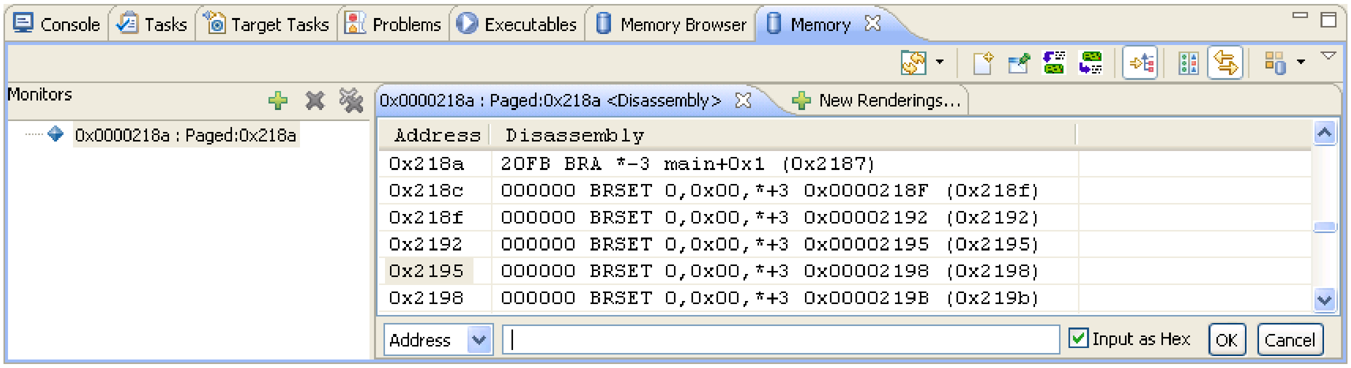
4.2.10.5 Go to Address

The memory view provides graphical controls to display memory at a specific address. To go to a specific address, perform these steps.

1. Open the Memory view.
2. In the Renderings pane, select the tab that corresponds to the disassembly rendering for which you want to display a specific address.
3. Right-click in the Renderings pane and select Go to Address from the context menu.

   A group of controls appears on the Renderings pane.

4. In the blank text box, enter the address that you want to display.



Address	Disassembly
0x218a	20FB BRA *-3 main+0x1 (0x2187)
0x218c	000000 BRSET 0,0x00,*+3 0x0000218F (0x218f)
0x218f	000000 BRSET 0,0x00,*+3 0x00002192 (0x2192)
0x2192	000000 BRSET 0,0x00,*+3 0x00002195 (0x2195)
0x2195	000000 BRSET 0,0x00,*+3 0x00002198 (0x2198)
0x2198	000000 BRSET 0,0x00,*+3 0x0000219B (0x219b)

Address | ☑ Input as Hex | OK | Cancel

Figure 4-42. Disassembly Rendering - Go to Address

NOTE

Check the Input as Hex checkbox only if you enter the address in hexadecimal notation.

5. Click OK to have the Disassembly rendering scroll to the specified address. Alternatively, click Cancel to abort the operation and hide the group of controls.

4.2.11 Hard Resetting

Use the reset hard command in the Debugger Shell view to send a hard reset signal to the target processor.

NOTE

The Hard Reset command is enabled only if the debug hardware you are using supports it.

4.3 Debugging Externally Built Executable Files

You can use the Microcontrollers ELF executable wizards to debug .elf files generated by a different IDE.

The main purpose of the **MCU Executable Importer** wizards is to create a CodeWarrior for Microcontrollers Eclipse project that can be readily debugged starting from an executable file build with a Microcontrollers toolchain.

The **MCU Executable Import** and **PA ELF Executable** wizards let you import a `*.elf`, `*.abs`, or `*.flt` file and associate it to a project.

This topic describes the various pages of the two wizards and the list steps that assists you in importing and associating an executable (`*.elf`, `*.abs`, or `*.flt`) file to a project.

- Microcontrollers ELF Executable
- Importing Projects from Command Line
- Debug an Externally Built Microcontrollers Executable File

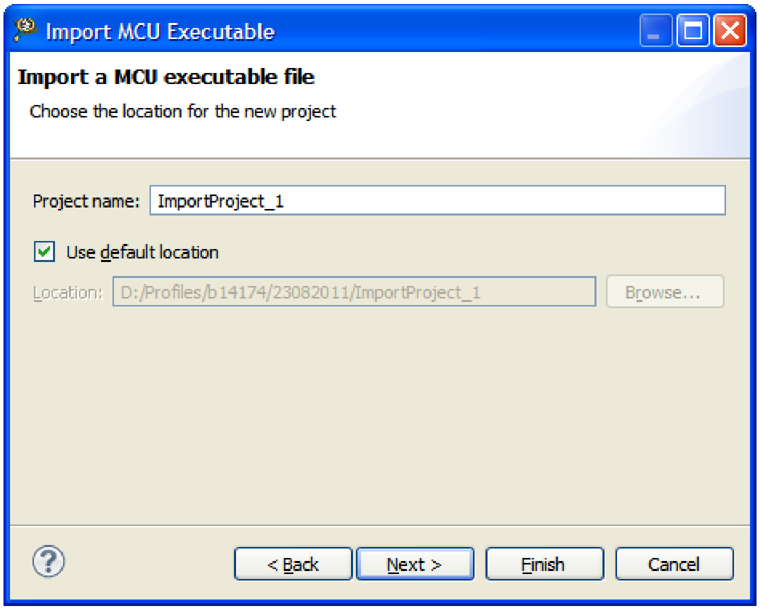
4.3.1 Microcontrollers ELF Executable

The pages in the Microcontrollers ELF Executable are:

- Import a MCU Executable File Page
- Select MCU executable file to be imported Page
- Device and Board Page
- Connections Page

4.3.1.1 Import a MCU Executable File Page

Use this page to name your new project, and specify the workspace directory.



Import MCU Executable

Import a MCU executable file

Choose the location for the new project

Project name: ImportProject_1

☑ Use default location

Location: D:/Profiles/b14174/23082011/ImportProject_1

Figure 4-43. Import MCU Executable - Import a MCU Executable File Page

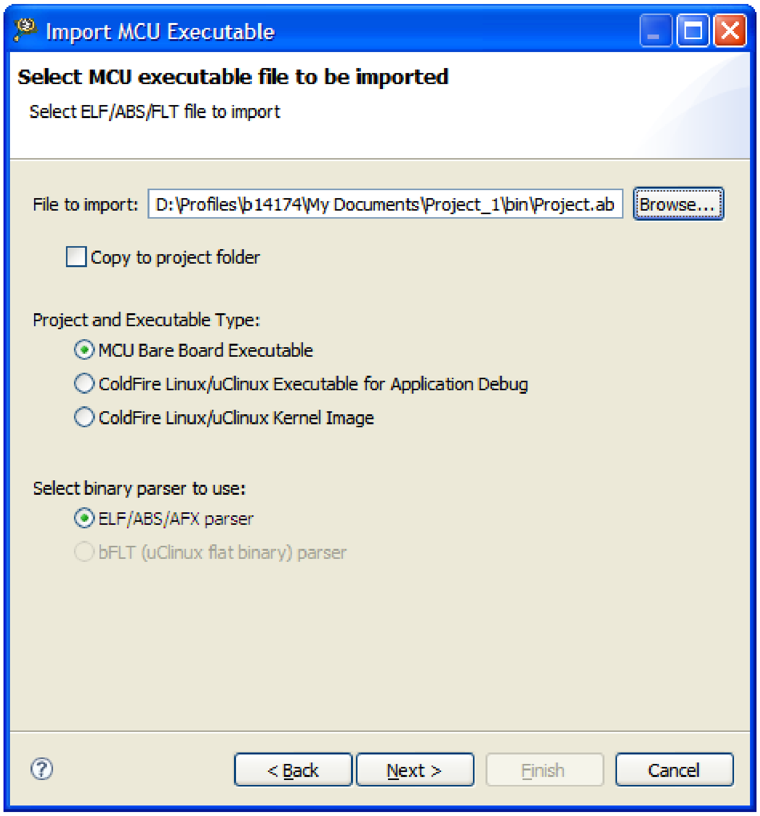
The following table describes the purpose of the various options.

Table 4-33. Import a MCU Executable File Page Settings

Option	Description
New Project Name	Enter the name for the new project in this text box.
Use default location	Stores the files required to build the program in the Workbench's current workspace directory. The project files are stored in the default location. Uncheck the Use default location checkbox and click Browse to specify a new location.
Location	Specifies the directory that contains the project files. Click Browse to navigate to the desired directory. This option is only available when Use default location checkbox is clear.
Choose file system	Specifies the file system to use. However, this option is available only when Use default location checkbox is clear. You can select either the default file system or a Target Management via Remote System Explorer (RSE).

4.3.1.2 Select MCU executable file to be imported Page

Use this page to select an existing Microcontrollers `ELF, ABS, or FLT` file you want to import.



Import MCU Executable

Select MCU executable file to be imported

Select ELF/ABS/FLT file to import

File to import: D:\Profiles\b14174\My Documents\Project_1\bin\Project.ab

□ Copy to project folder

Project and Executable Type:

● MCU Bare Board Executable

○ ColdFire Linux/uClinux Executable for Application Debug

○ ColdFire Linux/uClinux Kernel Image

Select binary parser to use:

● ELF/ABS/AFX parser

○ bFLT (uClinux flat binary) parser

Figure 4-44. Import MCU Executable - Select MCU executable file to be imported Page

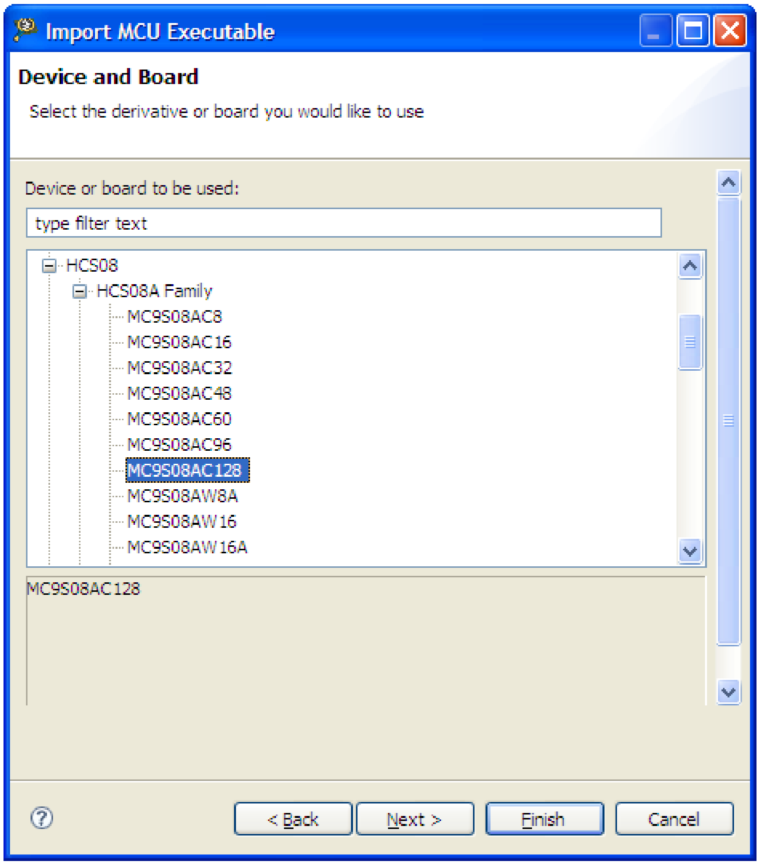
The following table explains the options available on the page.

Table 4-34. Select MCU executable file to be imported Page Settings

Option	Description
File to Import	Specifies the path of the ELF, ABS, or FLT files.
Browse	Click to locate the directory that contains the `*.elf`, `*.abs`, or `*.flt` files.
Copy to project	Check to copy the selected import file in the new project.
MCU Bareboard Executable	Select to use the microcontrollers bareboard executable.
ColdFire Linux/uClinux Executable for Application Debug	Select to use the ColdFire Linux/uClinux executable for application debug.
ColdFire Linux/uClinux Kernel Image	Select to use the ColdFire Linux/uClinux Kernel Image.
Select binary parser	Select a binary parser for the executable to be imported into the CodeWarrior IDE. The drop-down list includes various parsers supported by the CodeWarrior IDE.The commonly used parsers are: • ELF/ABS/AFX parser • bFLT (uClinux flat binary) parser

4.3.1.3 Device and Board Page

Use this page to select the derivative or board you would like to use.



Import MCU Executable

Device and Board

Select the derivative or board you would like to use

Device or board to be used:

type filter text

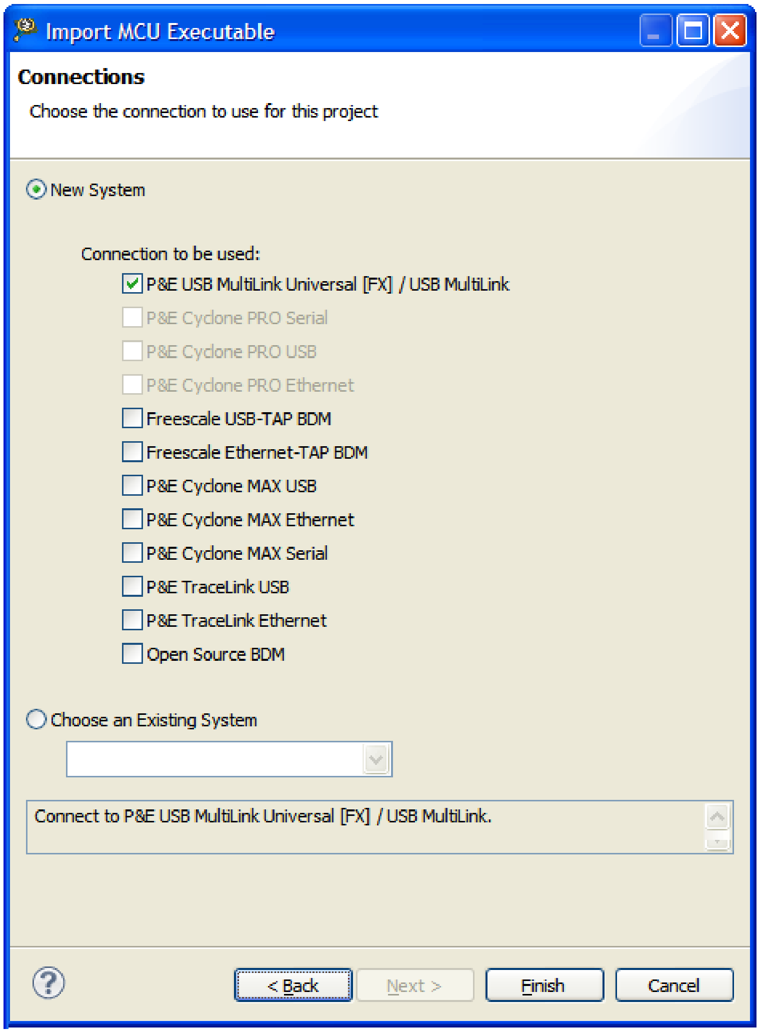
- HCS08
  - HCS08A Family
    - MC9S08AC8
    - MC9S08AC16
    - MC9S08AC32
    - MC9S08AC48
    - MC9S08AC60
    - MC9S08AC96
    - MC9S08AC128
    - MC9S08AW8A
    - MC9S08AW16
    - MC9S08AW16A

MC9S08AC128

Figure 4-45. Import MCU Executable - Devices Page

4.3.1.4 Connections Page

Use this page to select a connection to use for the project. Depending on the selected derivative or board, the connections will appear enabled or grayed out.



Import MCU Executable

Connections

Choose the connection to use for this project

● New System

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone PRO Serial
- □ P&E Cyclone PRO USB
- □ P&E Cyclone PRO Ethernet
- □ Freescale USB-TAP BDM
- □ Freescale Ethernet-TAP BDM
- □ P&E Cyclone MAX USB
- □ P&E Cyclone MAX Ethernet
- □ P&E Cyclone MAX Serial
- □ P&E TraceLink USB
- □ P&E TraceLink Ethernet
- □ Open Source BDM

○ Choose an Existing System

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 4-46. Import MCU Executable - Connections Page

4.3.2 Importing Projects from Command Line

To import some projects into a specific workspace from command line:

1. Copy the projects into the workspace location

2. Execute the command:

```
ecd -updateWorkspace -data your_workspace_location.
```

For example:

```
cwidec -application com.freescale.wsupdater.updater -data your_workspace_location
```

The following options are supported:

-logfile <file>

By default, the workspace updater outputs information to stdout. However, you can ask for the output to go to a file

```
-noclose
```

By default, the workspace updater closes the workbench after it has completed adding/removing projects. However, the user can request that the workbench remain open.

This application brings up the Eclipse IDE. The user is presented with a modal progress dialog while the update operation is in progress. The dialog is a standard Jobs dialog. It not only provides progress but gives the user the option to dismiss the dialog and have the operation run in the "background". Also, like most jobs in Eclipse, the operation is cancelable.

4.3.3 Debug an Externally Built Microcontrollers Executable File

You can use the Microcontrollers ELF Executable wizard to debug an .elf file generated by a different IDE.

To debug externally built executable files, perform these steps.

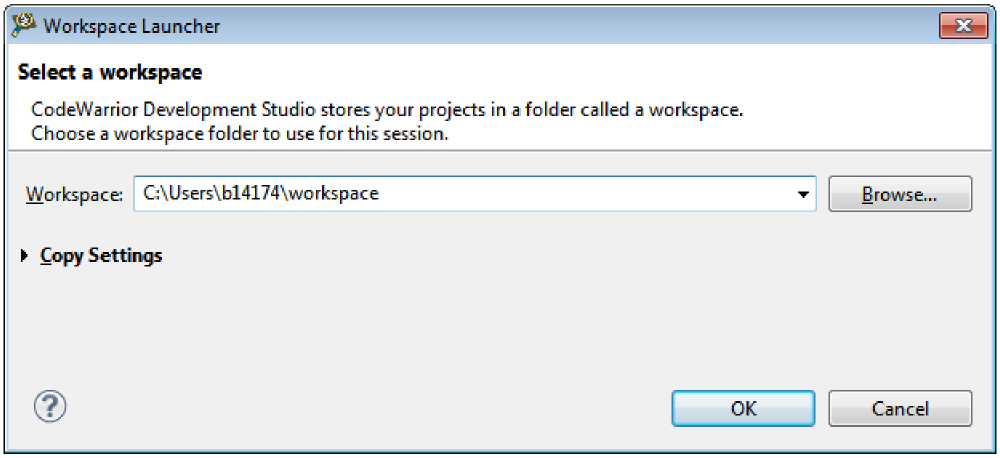
1. Import a MCU Executable File Page
2. Specify Executable File to Import
3. Select Derivative or Board
4. Select Connection
5. Edit Launch Configuration
6. Source Lookup Path
7. Debug Executable File

4.3.3.1 Import a MCU Executable File

You specify the externally built executable file that you want to debug in the CodeWarrior IDE. The IDE imports the executable file into a new project. To specify the executable file, perform these steps.

1. Select Start > Programs > Freescale CodeWarrior > **CW MCU V10.x** > CodeWarrior.

   The IDE launches and the WorkSpace Launcher dialog box prompts you to select a workspace to use.



Select a workspace

CodeWarrior Development Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\b14174\workspace

Figure 4-47. WorkSpace Launcher Dialog Box

2. Click OK to accept the default workspace. To use a workspace different from the default, click Browse and specify the desired workspace.

   The IDE starts and displays the Welcome page.

   NOTE

   You can also select the Use this as the default and do not ask again checkbox to set default/selected path as a default location for storing all your projects.

3. Click the **Go to Workbench** link.

   The **Workbench** window opens.

4. Select File > Import, from the IDE menu bar.

   The Import wizard appears.

5. Expand the CodeWarrior group.
6. Select MCU Executabl **e Importer** to debug a Microcontrollers `*.elf,` `*.abs,` or `*.flt` file.
7. Click Next.

   The Import a MCU executable file page appears.

8. Specify a name for the new project. For example, enter the project name as `ImportProject_1`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name.

Alternatively, click **checkbox. In** select the desired location from the **Browse For Folder** dialog box and click **OK** . Ensure that you append the path with the name of the project to create a new location for your project.

9. Click **Next** .

   The **Select MCU executable file to be imported** page appears.

4.3.3.2 Specify Executable File to Import

1. Click **Browse** .

   The **Select file** dialog box appears.

2. Navigate to the executable file that you want to import and click **Open** .

   The path of the selected file appears in the **File to import** text box.

3. Check the **Copy to project** checkbox if you want to copy the specified file in the new project. By default, the **Copy to project** checkbox is cleared.
4. From the **Select binary parser to use** , select the parser you want to use.
5. Click **Next** .

   The **Devices** page appears.

4.3.3.3 Select Derivative or Board

1. Expand the tree control and select the derivative or board you would like to use. For example, select `HCS08 > HCS08A Family > MC9S08AC128`.
2. Click Next.

   The **Connections** page appears.

4.3.3.4 Select Connection

1. Select the desired connection from the **Connection to be used** group. For example, check the **P&E Full Chip Simulation** checkbox.

   NOTE

   You can select multiple connections by checking appropriate checkboxes in the **Connections** page.

2. Click **Finish** .

   The **Import MCU Executable** window closes and the wizard creates a project according to your specifications. You can access the project from the **CodeWarrior Projects view** in the **Workbench** window.

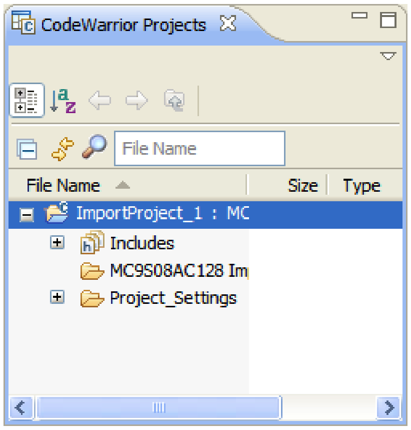


Figure 4-48. CodeWarrior Projects View

3. Right-click on the project and from the context menu select **Build Project** .

   The new project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

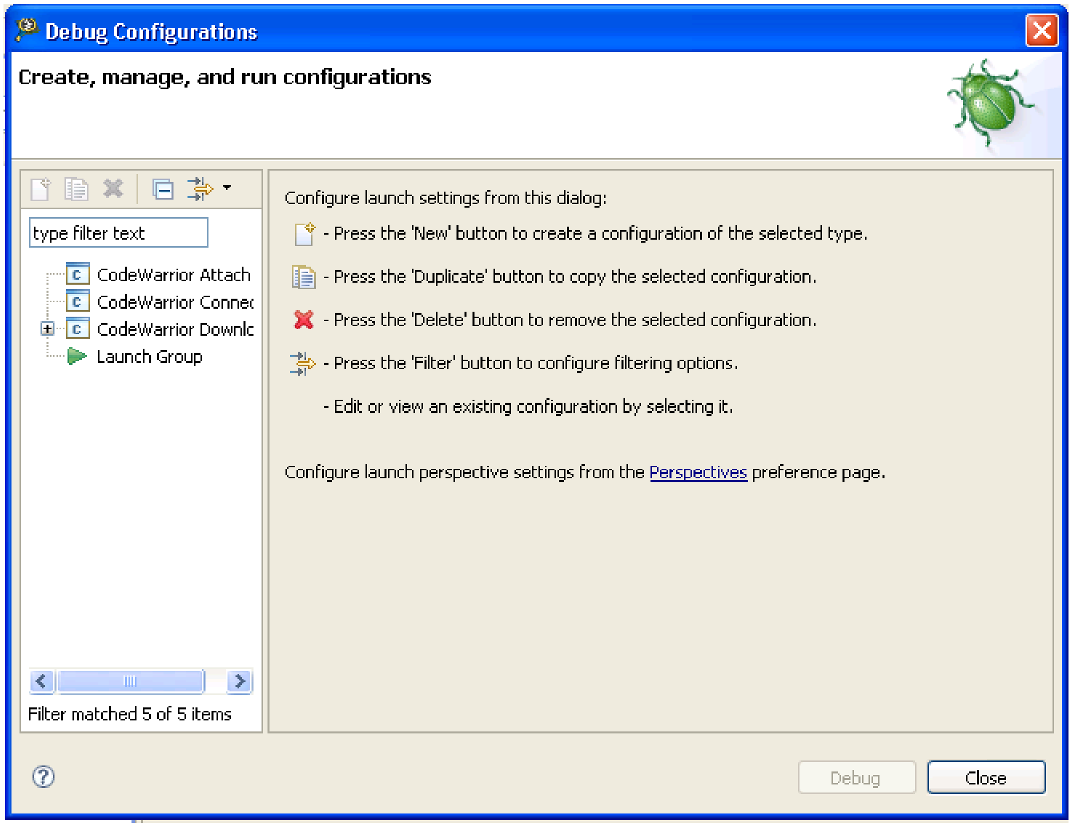
4.3.3.5 Edit Launch Configuration

Before you edit the launch configuration, ensure that you create a project for the executable file.

To edit the launch configuration for your executable file, perform these steps.

1. From the main menu bar of the IDE, select **Run > Debug Configurations** . The IDE uses the settings in the launch configuration to generate debugging information and initiate communications with the target board.

   The Debug Configurations dialog box appears. The left side of this dialog box has a list of debug configurations that apply to the current application.



Debug Configurations

Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.

- Press the 'Duplicate' button to copy the selected configuration.

- Press the 'Delete' button to remove the selected configuration.

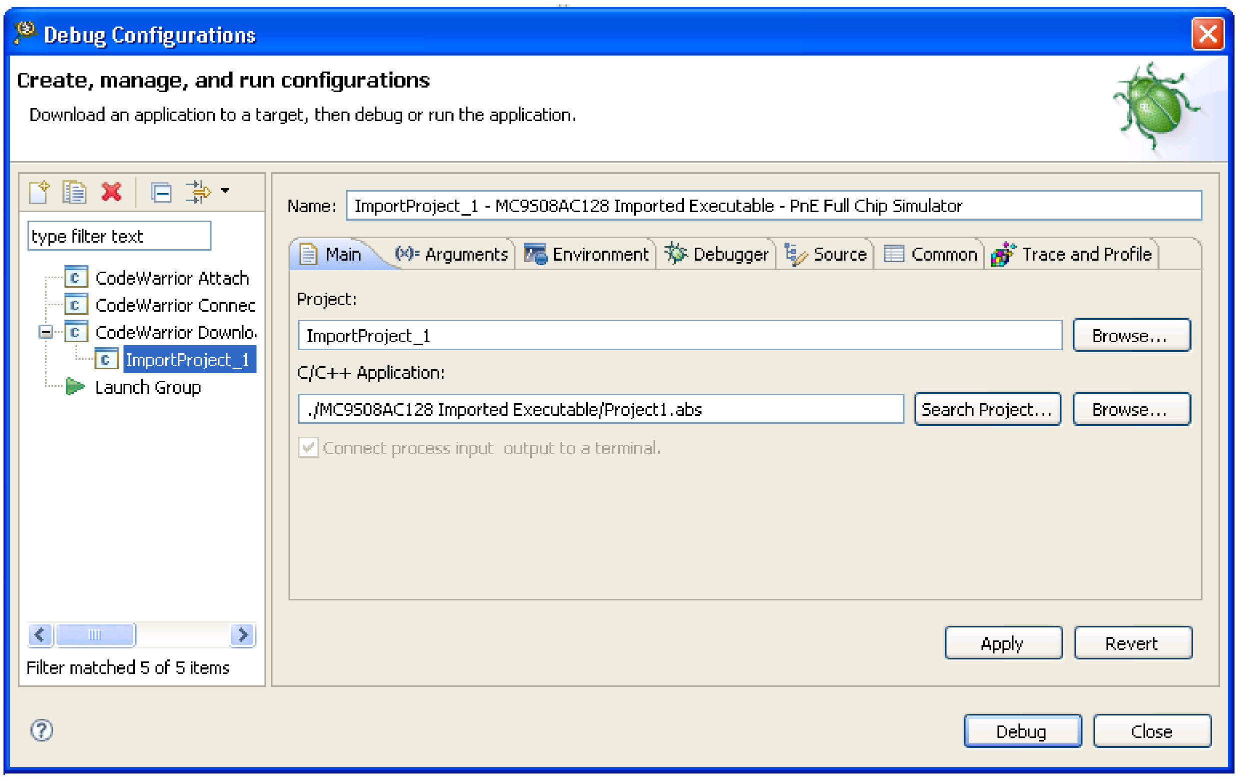
- Press the 'Filter' button to configure filtering options.

- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Figure 4-49. Debug Configurations Dialog Box

2. Expand the CodeWarrior Download configuration.
3. From the expanded list, select the newly created debug configuration. For example, select *ImportProject_1 - MC9S08AC128*.



Debug Configurations

Create, manage, and run configurations

Download an application to a target, then debug or run the application.

type filter text

- CodeWarrior Attach
- CodeWarrior Connec
- CodeWarrior Downlo.
  - ImportProject_1
- Launch Group

Filter matched 5 of 5 items

Name: ImportProject_1 - MC9S08AC128 Imported Executable - PnE Full Chip Simulator

Main | Arguments | Environment | Debugger | Source | Common | Trace and Profile

Project:

ImportProject_1

C/C++ Application:

./MC9S08AC128 Imported Executable/Project1.abs

☑ Connect process input output to a terminal.

Figure 4-50. Selected Launch Configuration

4. Click the Debugger tab of the Debug Configurations dialog box.

   The corresponding page appears.

5. Use the **Debugger** list box to specify the debugger that corresponds to your type of executable file.

6. Configure the debugger options as appropriate for your executable file.

   For example, specify the appropriate target processor, any initialization files, and connection protocol.

4.3.3.6 Source Lookup Path

You need to specify the source lookup path in terms of the compilation path and the local file-system path for the newly imported executable file. The CodeWarrior debugger uses both of these paths to debug the executable file.

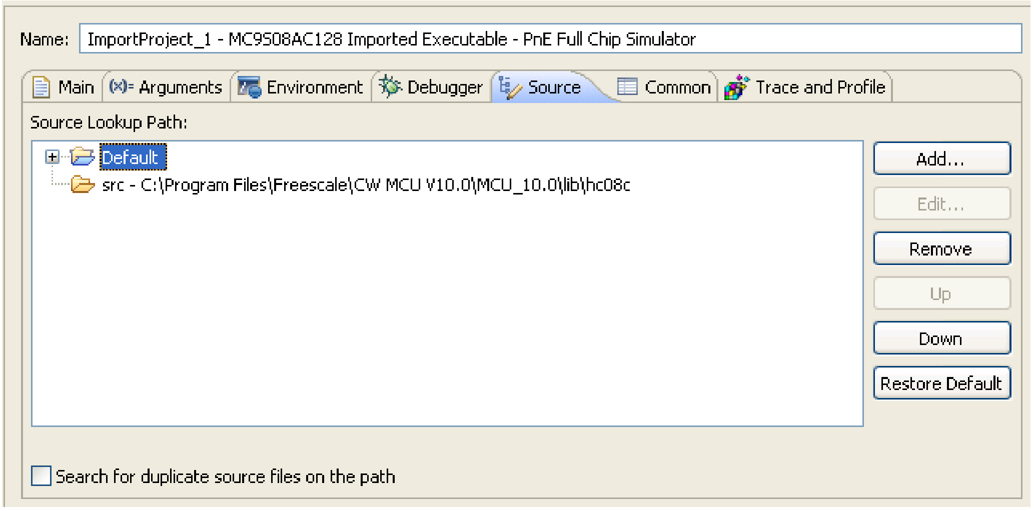
The compilation path is the path to the original project that built the executable file. If the original project is from an IDE on a different computer, you specify the compilation path in terms of the file system on that computer.

The local file-system path is the path to the project that the CodeWarrior IDE creates in order to debug the executable file.

To specify a source lookup path for your executable file:

1. Click the Source tab of the Debug Configurations dialog box.

   The corresponding page appears.



Name: ImportProject_1 - MC9S08AC128 Imported Executable - PnE Full Chip Simulator

Source Lookup Path:

- Default
  - src - C:\Program Files\Freescale\CW MCU V10.0\MCU_10.0\lib\hc08c

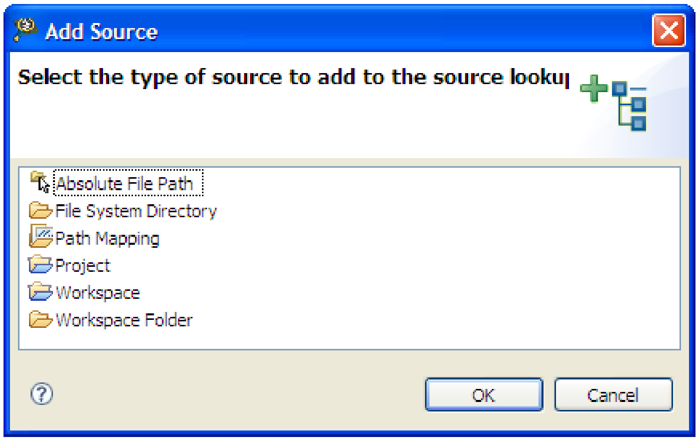
□ Search for duplicate source files on the path

Figure 4-51. Debug Configurations Dialog Box - Source Page

2. Click Add.

   The Add Source dialog box appears.

3. Select **Path Mapping**.



Select the type of source to add to the source looku

- Absolute File Path
- File System Directory
- Path Mapping
- Project
- Workspace
- Workspace Folder

Figure 4-52. Add Source Dialog Box

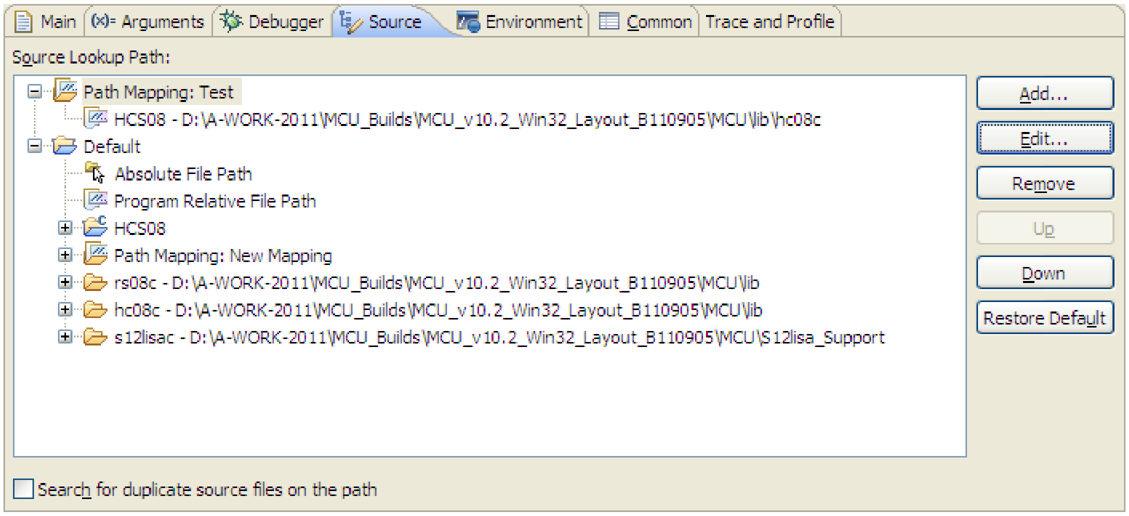
4. Click OK.

   The **Path Mappings** dialog box appears.

5. Provide a name for the new path mapping.
6. Specify a name for the compilation path.

7. In the **Local file system path** text box, enter the path to the parent project of the executable file, relative to your computer. Alternatively, click the ellipis button to specify the parent project.
8. Click **OK** .

   The Path Mappings dialog box closes. The IDE selects the new mapping in the **Source Lookup Path** list of the Source page.



Source Lookup Path:

```
Path Mapping: Test
    HCS08 - D:\A-WORK-2011\MCU_Builds\MCU_v10.2_Win32_Layout_B110905\MCU\lib\hc08c
Default
    Absolute File Path
    Program Relative File Path
    HCS08
    Path Mapping: New Mapping
    rs08c - D:\A-WORK-2011\MCU_Builds\MCU_v10.2_Win32_Layout_B110905\MCU\lib
    hc08c - D:\A-WORK-2011\MCU_Builds\MCU_v10.2_Win32_Layout_B110905\MCU\lib
    s12lisac - D:\A-WORK-2011\MCU_Builds\MCU_v10.2_Win32_Layout_B110905\MCU\S12lisa_Support
```

☐ Search for duplicate source files on the path

Figure 4-53. Source Lookup Path

Tip

You can use the IDE to discover the path to the parent project of the executable file, relative to the computer that generated the file. In the **C/C++ Projects** view of the **C/C++** perspective, expand the project that contains the executable file that you want to debug. Next, expand the group that has the name of the executable file itself. A list of paths appears, relative to the computer that generated the file. Search this list for the names of source files used to build the executable file. The path to the parent project of one of these source files is the path you should enter in the **Compilation path** text box.

9. If required, change the order in which the IDE searches the paths.

   The IDE searches the paths in the order shown in the **Source Lookup Path** list, stopping at the first match. To change this order, select a path, then click the Up or Down button to change its position in the list.

10. Click Apply.

The IDE saves your changes.

4.3.3.7 Debug Executable File

Use the CodeWarrior debugger to debug the externally built executable file.

To debug the executable file, click the Debug button of the Debug Configurations dialog box.

Chapter 5
Kinetis Cache Viewer

This chapter describes the Kinetis Cache Viewer, which is available in CodeWarrior for Microcontrollers, Version 10.2. This feature works only for Kinetis derivatives that have cache, as Pioneer 3. In this chapter:

- Kinetis Cache
- CodeWarrior Cache View for Kinetis
- Performance Considerations and Kinetis Particularities

5.1 Kinetis Cache

There are two caches available in a Pioneer 3 MCU. The *Code Cache*, which is associated with the Processor Code (PC) Bus and the *System Cache*, associated the Processor System (PS) bus. One important aspect here is that both are unified caches, meaning that both are capable of storing instructions and data as well. However, as the bus names imply, the typical operations issue code accesses on the PC bus and the data accesses on the PS bus. The buses are partitioned as follows:

- The memory accesses to the addresses from `0x0` to `0x1FFFFFFF` use the **PC** bus.
- The memory accesses to the addresses from `0x20000000` to `0xFFFFFFFF` use the **PS** bus.

The figure below describes the Kinetis P3 local memory controller.

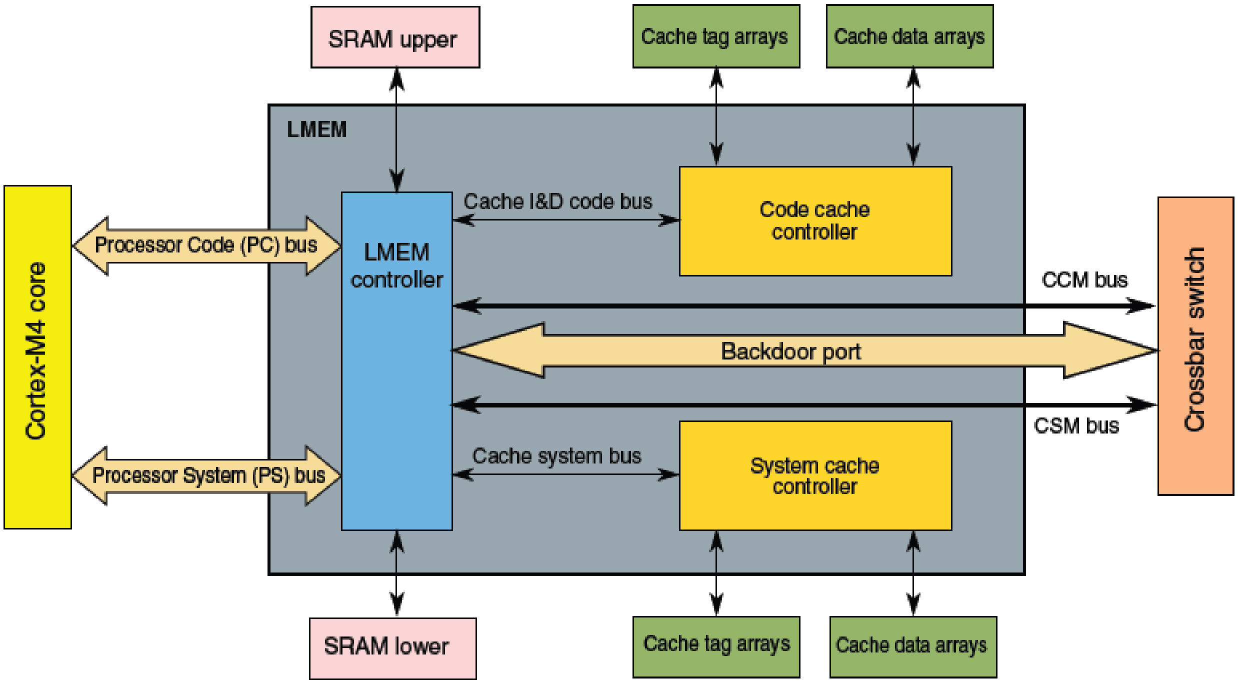


Figure 5-1. Kinetis P3 Local Memory Controller Block Diagram

In terms of structure, the *Code Cache* and the *System Cache* are identical. Each of them has 8KB, they are 2-way set associative and have *four* word lines (16 bytes).

Whenever a cache address is targeted to be processed by the cache module, its address generates the *Tag* and the *Cache Line* in a way, that the upper 20 bits represent the *Tag* and the next 8 bytes represent the *Line*. The bits 3-2 determine the word within a line and bits 1-0 determine the byte within a word.

31 12	11 4	3 2	1 0
Tag	Line Number	W	B

Figure 5-2. Tag and Line Number in a 32 Bits Address on Kinetis

Whenever a cache-miss occurs, one line data consisting in 16 bytes will be stored into the cache in the cache data array at the line number index at one available way. In addition the line tag will be stored into the cache tag array at the line number index.

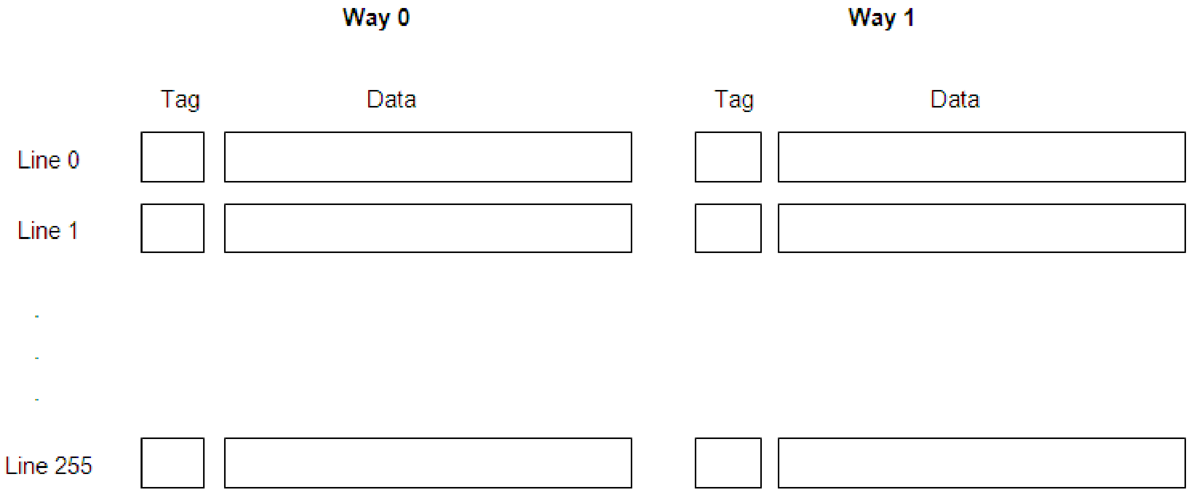


Figure 5-3. Kinetis Cache Contents

Each cache line has the following properties:

- **Valid** - the cache line contains valid data that can be used instead of the corresponding data from the memory
- **Modified (Dirty)** - the cache line is modified by the core, but its content is not saved back to the corresponding memory yet.

The Kinetis local memory controller supports three modes of operation:

- Write-through
- Write-back
- Non-cacheable

5.1.1 Write-through

The write-through operation inculdes:

- A write-through read miss on the input bus causes a line read on the output bus of a 16-byte-aligned memory address containing the desired address. This data is loaded into the cache and is marked as valid and not modified.
- A write-through read hit to a valid cache location returns data from the cache with no output bus access.

- A write-through write miss bypasses the cache and writes to the output bus without allocating and fetching the corresponding line into the cache (no allocate on write miss policy)
- A write-through write hit updates the cache hit data and writes to the output bus.

5.1.2 Write-back

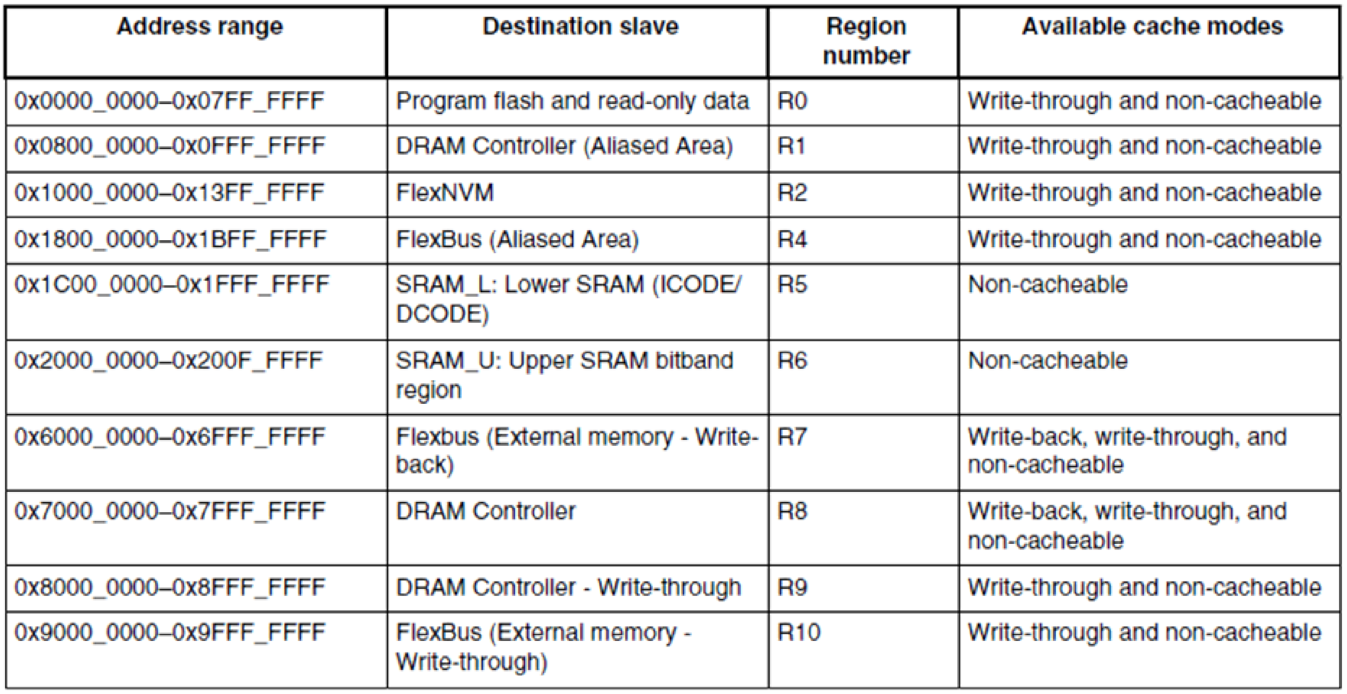
The write-back operation inculdes:

- A write-back read miss on the input bus causes a line read on the output bus of a 16-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and marked as valid and not modified.
- A write-back read hit to a valid cache location returns data from the cache with no output bus access.
- A write-back write miss brings the corresponding line into the cache (allocate on write miss policy). If the line to be brought into the cache, replaces another dirty line, the latter is saved back to the memory before reading the new line. In this case, a line read on the output bus of a 16 byte aligned memory address containing the desired write address is performed.

5.1.3 Non-cacheable

The non-cacheable accesses bypass the cache and access the output bus.

The Kinetis memory map contains multiple regions as shown in the following table. Each region has one or more available cache modes.



Address range	Destination slave	Region number	Available cache modes
0x0000_0000–0x07FF_FFFF	Program flash and read-only data	R0	Write-through and non-cacheable
0x0800_0000–0x0FFF_FFFF	DRAM Controller (Aliased Area)	R1	Write-through and non-cacheable
0x1000_0000–0x13FF_FFFF	FlexNVM	R2	Write-through and non-cacheable
0x1800_0000–0x1BFF_FFFF	FlexBus (Aliased Area)	R4	Write-through and non-cacheable
0x1C00_0000–0x1FFF_FFFF	SRAM_L: Lower SRAM (ICODE/ DCODE)	R5	Non-cacheable
0x2000_0000–0x200F_FFFF	SRAM_U: Upper SRAM bitband region	R6	Non-cacheable
0x6000_0000–0x6FFF_FFFF	Flexbus (External memory - Write-back)	R7	Write-back, write-through, and non-cacheable
0x7000_0000–0x7FFF_FFFF	DRAM Controller	R8	Write-back, write-through, and non-cacheable
0x8000_0000–0x8FFF_FFFF	DRAM Controller - Write-through	R9	Write-through and non-cacheable
0x9000_0000–0x9FFF_FFFF	FlexBus (External memory - Write-through)	R10	Write-through and non-cacheable

Figure 5-4. Kinetis Cache Regions

The Kinetis architecture supports the following cache operations:

- **Read** - The cache data and tag are accessible for reading.
- **Write** - The user is able to modify the cache data using the Local Memory Controller support.
- **Invalidate** - Unconditionally clear, valid and modify bits of a cache entry.
- **Push (or Synchronize)** - Writes a cache entry in the memory if it is valid and modified, then clear the modify bit. If entry not valid or not modified, leave as is.
- **Clear (or Flush)** - Push a cache entry if it is valid and modified, then clear the valid and modify bits. If the entry is not valid or not modified, clear the valid bit.

The scope of *Invalidate*, *Push* and *Clear* commands may be a particular cache line or an entire cache way.

5.2 CodeWarrior Cache View for Kinetis

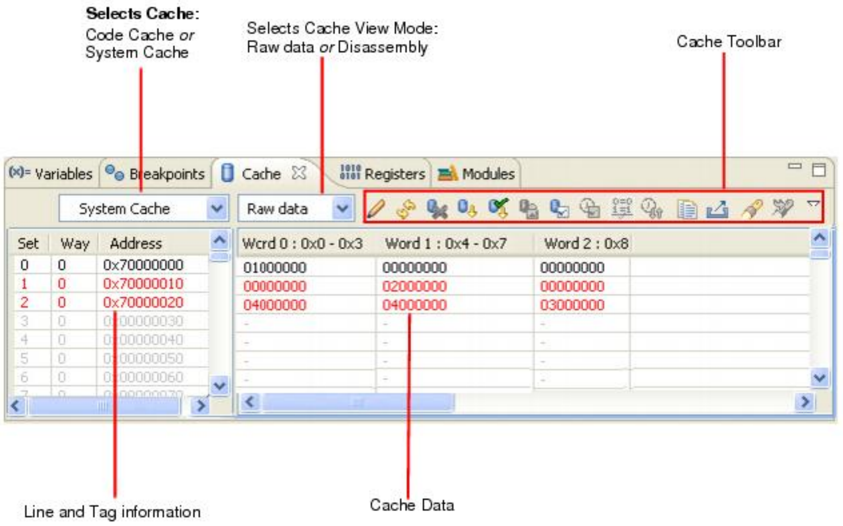
To open the **Cache** View:

1. Start a debugging session.
2. In the **Debug** view of the **Debug** perspective, select the process for which you want to work with Cache.
3. Select **Window > Show View > Other** .

The **Show View** dialog box appears.

4. Expand the **Debug** tree control.
5. Select **Cache.**
6. Click **OK** .

The **Show View** dialog box closes. The **Cache** view appears, attached to an existing collection of views in the current perspective.



System Cache

Raw data

Set	Way	Address
0	0	0x70000000
1	0	0x70000010
2	0	0x70000020
3	0	0x70000030
4	0	0x70000040
5	0	0x70000050
6	0	0x70000060
7	0	0x70000070

Word 0 : 0x0 - 0x3	Word 1 : 0x4 - 0x7	Word 2 : 0x8
01000000	00000000	00000000
00000000	02000000	00000000
04000000	04000000	03000000
-	-	-
-	-	-
-	-	-
-	-	-

Figure 5-5. Cache View Structure

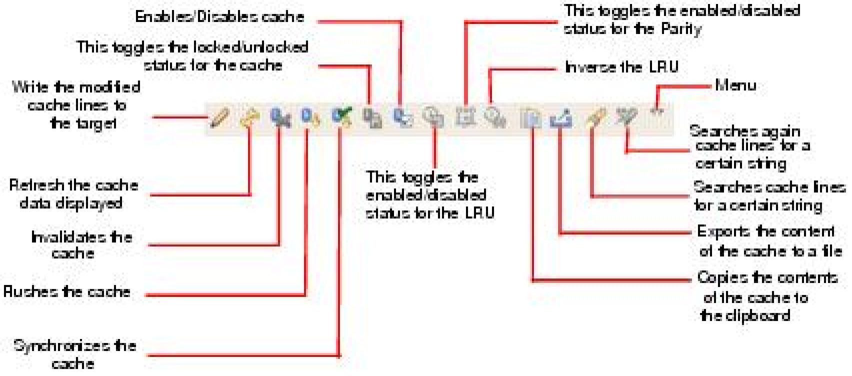

Enables/Disables cache
This toggles the locked/unlocked status for the cache
Write the modified cache lines to the target
Refresh the cache data displayed
Invalidates the cache
Flushes the cache
Synchronizes the cache
This toggles the enabled/disabled status for the LRU
This toggles the enabled/disabled status for the Parity
Inverse the LRU
Menu
Searches again cache lines for a certain string
Searches cache lines for a certain string
Exports the content of the cache to a file
Copies the contents of the cache to the clipboard

Figure 5-6. Cache View Toolbar

Line scope commands are available in the contextual menu, by right-clicking on a cache line (data or tag area). See the following picture.

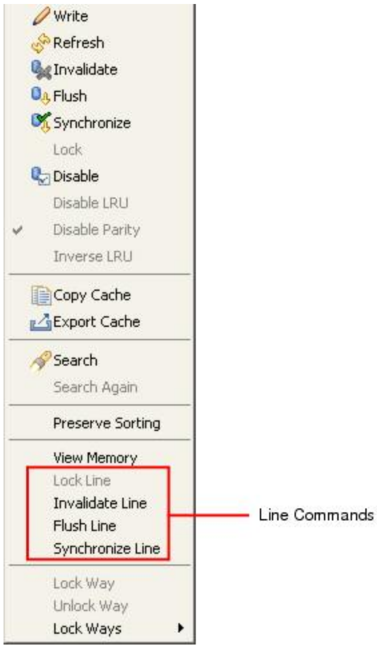


Figure 5-7. Cache View Context-menu

5.3 Performance Considerations and Kinetis Particularities

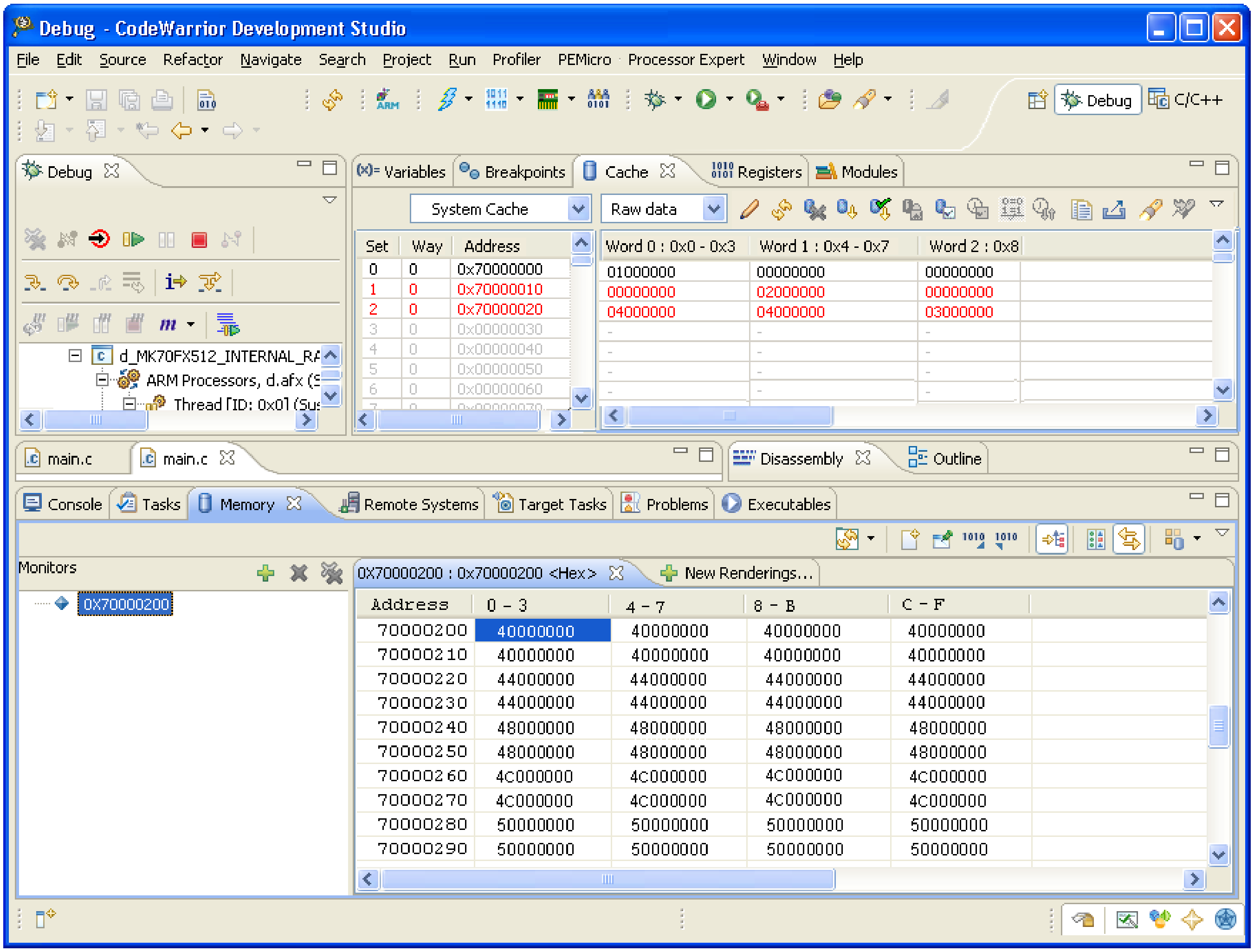
Kinetis local memory controller has some particularities that affect the debugger performance. Besides the common performance degradation given by the latency of reading the cache and updating the cache view, there are some specific performance issues that must be taken into account when performing various operations with the debugger.

When accessing a memory location, the Kinetis platform uses the same mechanism regardless the source of the access, the core or the debugger module. Therefore, without any preventive actions, the Kinetis cache may be easily polluted with debugger accesses to the memory (reads or writes). In order to solve this problem, additional algorithms were developed to avoid the debugger intrusiveness and also to maintain the cache coherency.

For example, the biggest performance degradation may be observed when stepping having the cache enabled and a memory monitor for a Write-back region active. At each step, besides the time required to populate the cache view, the user may experience a bigger latency of about 4 times longer. This latency is the result of the debugger intrusiveness prevention algorithm that is activated when accessing cacheable regions.

The algorithm is so slow because it performs cache searches for all the data that is read or written by the debugger. A workaround for this performance issue is to flush and then deactivate the cache before stepping with a write-back zone memory monitor active.

Another Kinetis specific issue is that the cache is not automatically flushed or cleared at a normal reset. This is done only at a Power-On reset. Therefore, if the user wants to use the cache in a debug session by enabling it using the **Cache** Viewer, the user shall also perform an *Invalidate* command right after the *Enable Cache* command.


Debug - CodeWarrior Development Studio
File Edit Source Refactor Navigate Search Project Run Profiler PEMicro Processor Expert Window Help
Debug
C/C++
Debug
d_MK70FX512_INTERNAL_RA
ARM Processors, d.afx (S
Thread [ID: 0x0] (Sus
Variables Breakpoints Cache Registers Modules
System Cache
Raw data
main.c
main.c
Disassembly Outline
Console Tasks Memory Remote Systems Target Tasks Problems Executables
Monitors
0X70000200
0X70000200 : 0x70000200 <Hex>
New Renderings...

Set	Way	Address	Word 0 : 0x0 - 0x3	Word 1 : 0x4 - 0x7	Word 2 : 0x8
0	0	0x70000000	01000000	00000000	00000000
1	0	0x70000010	00000000	02000000	00000000
2	0	0x70000020	04000000	04000000	03000000
3	0	0x00000030	-	-	-
4	0	0x00000040	-	-	-
5	0	0x00000050	-	-	-
6	0	0x00000060	-	-	-

Address	0 - 3	4 - 7	8 - B	C - F
70000200	40000000	40000000	40000000	40000000
70000210	40000000	40000000	40000000	40000000
70000220	44000000	44000000	44000000	44000000
70000230	44000000	44000000	44000000	44000000
70000240	48000000	48000000	48000000	48000000
70000250	48000000	48000000	48000000	48000000
70000260	4C000000	4C000000	4C000000	4C000000
70000270	4C000000	4C000000	4C000000	4C000000
70000280	50000000	50000000	50000000	50000000
70000290	50000000	50000000	50000000	50000000

Figure 5-8. Debug Perspective - Cache and Memory View

Chapter 6
Multicore Debugging

This chapter lists the steps to define multiple, arbitrary groupings of cores and perform multicore operations. This is of particular importance in the case that the hardware has provided a means to synchronize an operation across multiple cores.

Additionally, the chapter lists the steps to add multicore operations to Eclipse through both the UI and through the Debugger Shell. The operations are divided into two sets. One set is for controlling target execution state, also known as run control, and consists of Resume, Suspend and Step Into. The other set is for controlling debug session lifetime, which we'll refer to as session control, and consists of Restart and Terminate. An integral part of the feature will be the ability to select any subset of cores for operation.

The topic in this chapter is:

- Creating DPM/LSM Projects
- Debugging DPM/LSM Projects
- Debugging Multicore Project
- Editing Multicore Groups
- Editing Target Types

NOTE

CodeWarrior for Microcontrollers does not enable user debug subsequently core by core manually. This means when a debug session is started already, no other debug session is permitted.

6.1 Creating DPM/LSM Projects

The Power Architecture e200 Lock-Step Mode (LSM) and Dual Processor Mode (DPM) projects are ideal examples of a multicore project.

- Creating LSM Project
- Creating DPM Project

MPC5643L devices can operate in two modes of operation:

- **Lock-Step Mode (LSM) -** This mode takes its name from the execution of the same commands by both cores in synchronicity (lock-step).
- **Decoupled Parallel Mode (DPM) -** In this mode, each CPU core and connected channel run independently from the other one and redundancy checkers (RCCU) are disabled. MPC5643L devices support only static configuration at power-on (either LS or DP modes).

NOTE

One of the two modes is statically selected at power-up. The selected mode may be changed only going through a full power-on reset. Each of these modes has several specific submodes that the device can enter. These modes differ, for example, in the list of enabled modules, pin configurations, reset phase, and safety status.

Lock-Step Mode (LSM) is used to increase safety, Dual Processor Mode (DPM) is used to increase performance. 1 Device is in Lockstep Mode (LSM) and 0 Device is in Dual Processor Mode (DPM).

In the target processor, LSM/DPM configuration is written in the internal Flash Memory and therefore is persistent throughout successive debug sessions. A project created for one mode cannot be debugged on a target configured with the other mode, e.g. a LSM project cannot be debugged on a processor that was configured for DPM. Switch Between Decoupled Parallel and Lock-Step Modes describes how to change the processor configuration.

6.1.1 Creating LSM Project

To create a LSM, perform these steps.

1. Select **Start > Programs > Freescale CodeWarrior > CW MCU V10.x > CodeWarrior**.

The IDE launches and the **WorkSpace Launcher** dialog box prompts you to select a workspace to use.

2. Click **OK** to accept the default workspace. To use a workspace different from the default, click **Browse** and specify the desired workspace.

   The IDE starts and displays the **Welcome** page.

   NOTE

   You can also select the **Use this as the default and do not ask again** checkbox to set default/selected path as a default location for storing all your projects.

3. Click the **Go to Workbench** link.

   The **Workbench** window opens.

4. Select **File > New > Bareboard Project**, from the IDE menu bar.

   The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

5. Specify a name for the new project. For example, enter the project name as `Project_1`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK** . Ensure that you append the path with the name of the project to create a new location for your project.

6. Click **Next**.

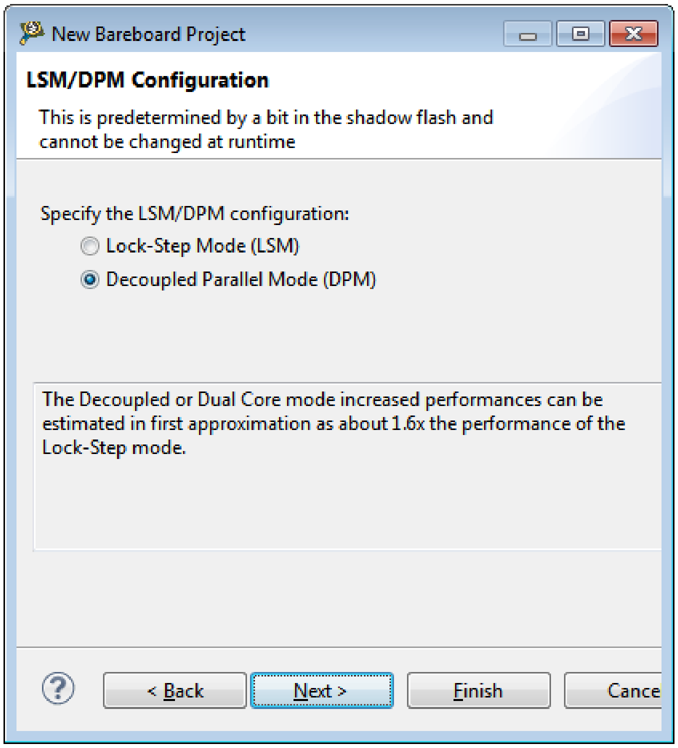
   The **Devices** page appears.

7. Expand the tree control and select the derivative or board you would like to use. For example, select **Qorivva > MPC56xxL Family > MPC5643L** .
8. Click **Next**.

   The **Connections** page appears.

9. Check the appropriate connection.

   The **LSM/ DPM Configuration** page appears.



This is predetermined by a bit in the shadow flash and cannot be changed at runtime

Specify the LSM/DPM configuration:

○ Lock-Step Mode (LSM)

● Decoupled Parallel Mode (DPM)

The Decoupled or Dual Core mode increased performances can be estimated in first approximation as about 1.6x the performance of the Lock-Step mode.

Figure 6-1. New Bareboard Project Wizard - LSM/ DPM Configuration Page

10. Select the **Lock-Step Mode (LSM)** option .
11. Click **Next**.

    The **Languages and Build Tools Options** page appears.

12. Select the appropriate options to enable programming language, build tools options, and floating point supports.
13. Click **Finish**.

    The wizard creates a project for the Kinetis architecture. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

6.1.2 Creating DPM Project

To create a DPM, perform these steps.

1. Select **Start > Programs > Freescale CodeWarrior > CW MCU V10.x > CodeWarrior**.

   The IDE launches and the **WorkSpace Launcher** dialog box prompts you to select a workspace to use.

2. Click **OK** to accept the default workspace. To use a workspace different from the default, click **Browse** and specify the desired workspace.

   The IDE starts and displays the **Welcome** page.

   NOTE

   You can also select the **Use this as the default and do not ask again** checkbox to set default/selected path as a default location for storing all your projects.

3. Click the **Go to Workbench** link.

   The **Workbench** window opens.

4. Select **File > New > Bareboard Project**, from the IDE menu bar.

   The **Create an MCU bareboard Project** page of the **New Bareboard Project** wizard appears.

5. Specify a name for the new project. For example, enter the project name as `Project_2`.

   NOTE

   If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name. Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK** . Ensure that you append the path with the name of the project to create a new location for your project.

6. Click **Next**.

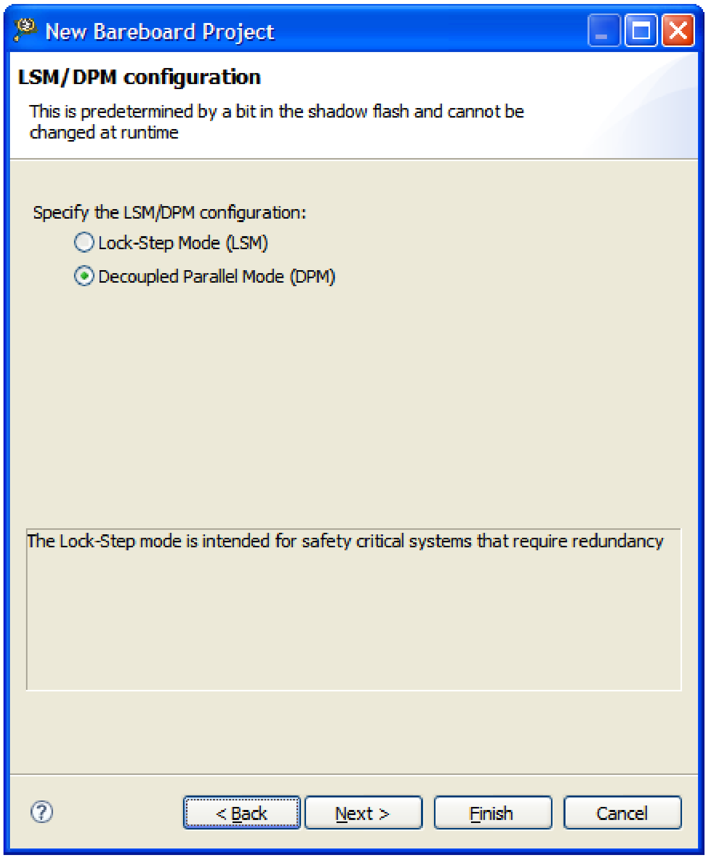
   The **Devices** page appears.

7. Expand the tree control and select the derivative or board you would like to use. For example, select **Qorivva > MPC56xxL Family > MPC5643L** .
8. Click **Next**.

   The **Connections** page appears.

9. Check the appropriate connection.

   The **LSM/ DPM Configuration** page appears.



LSM/ DPM configuration

This is predetermined by a bit in the shadow flash and cannot be changed at runtime

Specify the LSM/DPM configuration:

○ Lock-Step Mode (LSM)

● Decoupled Parallel Mode (DPM)

The Lock-Step mode is intended for safety critical systems that require redundancy

Figure 6-2. New Bareboard Project Wizard - LSM/ DPM Configuration Page

10. Select the **Decoupled Parallel Mode (DPM)** option.
11. Click **Next**.

    The **Languages and Build Tools Options** page appears.

12. Select the appropriate options to enable programming language, build tools options, and floating point supports.
13. Click **Finish**.

    The wizard creates a project for the Kinetis architecture. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

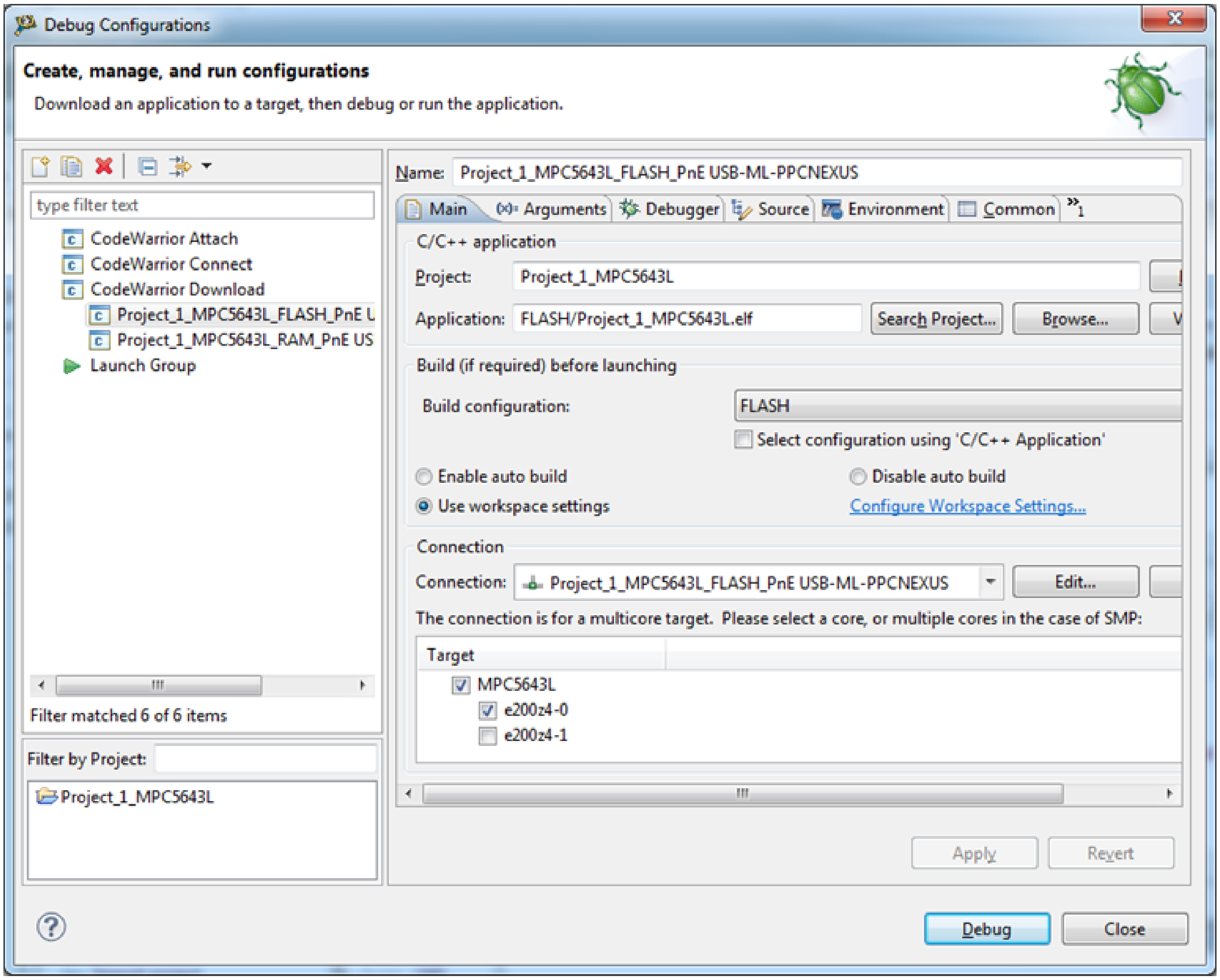
6.2 Debugging DPM/LSM Projects

The settings in the launch configurations handles all core-specific initializations. To set up the launch configurations, perform these steps.

1. Ensure that you machine has USB NEXUS Multilink connected to a Leopard (MPC5643L) DPM board.

2. Select **Run > Debug Configurations**.

   A menu of launch configurations appears.



Debug Configurations

Create, manage, and run configurations

Download an application to a target, then debug or run the application.

type filter text

- CodeWarrior Attach
- CodeWarrior Connect
- CodeWarrior Download
  - Project_1_MPC5643L_FLASH_PnE U
  - Project_1_MPC5643L_RAM_PnE US
- Launch Group

Filter matched 6 of 6 items

Filter by Project:

Project_1_MPC5643L

Name: Project_1_MPC5643L_FLASH_PnE USB-ML-PPCNEXUS

C/C++ application

Project: Project_1_MPC5643L

Application: FLASH/Project_1_MPC5643L.elf

Build (if required) before launching

Build configuration: FLASH

□ Select configuration using 'C/C++ Application'

○ Enable auto build

● Use workspace settings

○ Disable auto build

Configure Workspace Settings...

Connection

Connection: Project_1_MPC5643L_FLASH_PnE USB-ML-PPCNEXUS

The connection is for a multicore target. Please select a core, or multiple cores in the case of SMP:

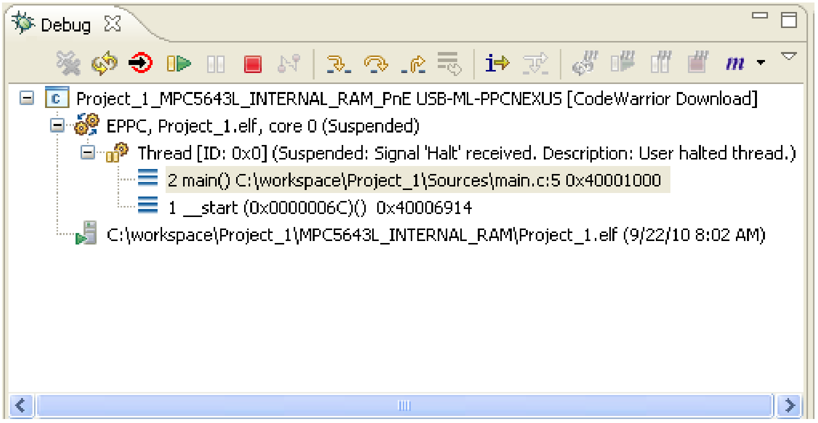
Target

- ☑ MPC5643L
  - ☑ e200z4-0
  - □ e200z4-1

Figure 6-3. Debug Configurations Dialog Box

3. Select the core or cores to be debugged using the Target checkboxes. In this case you can debug core 0 or both cores.
4. Click **Debug**.
5. Debugger halts execution at first statement of main().

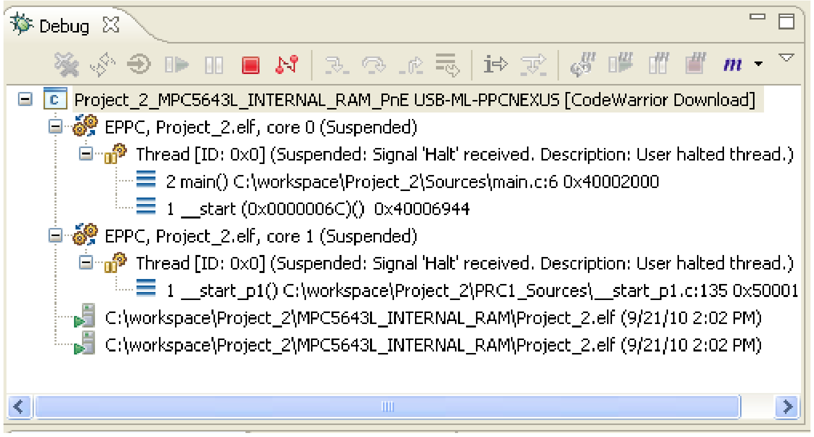
   The **Debug** view displays all the threads associated with the core.



```
Project_1_MPC5643L_INTERNAL_RAM_PnE USB-ML-PPCNEXUS [CodeWarrior Download]
    EPPC, Project_1.elf, core 0 (Suspended)
        Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
            2 main() C:\workspace\Project_1\Sources\main.c:5 0x40001000
            1 __start (0x0000006C)() 0x40006914
    C:\workspace\Project_1\MPC5643L_INTERNAL_RAM\Project_1.elf (9/22/10 8:02 AM)
```

Figure 6-4. Multicore Debugging - Debug Core 0

6. Repeat steps one through four to download all the other cores.



```
Project_2_MPC5643L_INTERNAL_RAM_PnE USB-ML-PPCNEXUS [CodeWarrior Download]
    EPPC, Project_2.elf, core 0 (Suspended)
        Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
            2 main() C:\workspace\Project_2\Sources\main.c:6 0x40002000
            1 __start (0x0000006C)() 0x40006944
    EPPC, Project_2.elf, core 1 (Suspended)
        Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
            1 __start_p1() C:\workspace\Project_2\PRC1_Sources\__start_p1.c:135 0x50001
    C:\workspace\Project_2\MPC5643L_INTERNAL_RAM\Project_2.elf (9/21/10 2:02 PM)
    C:\workspace\Project_2\MPC5643L_INTERNAL_RAM\Project_2.elf (9/21/10 2:02 PM)
```

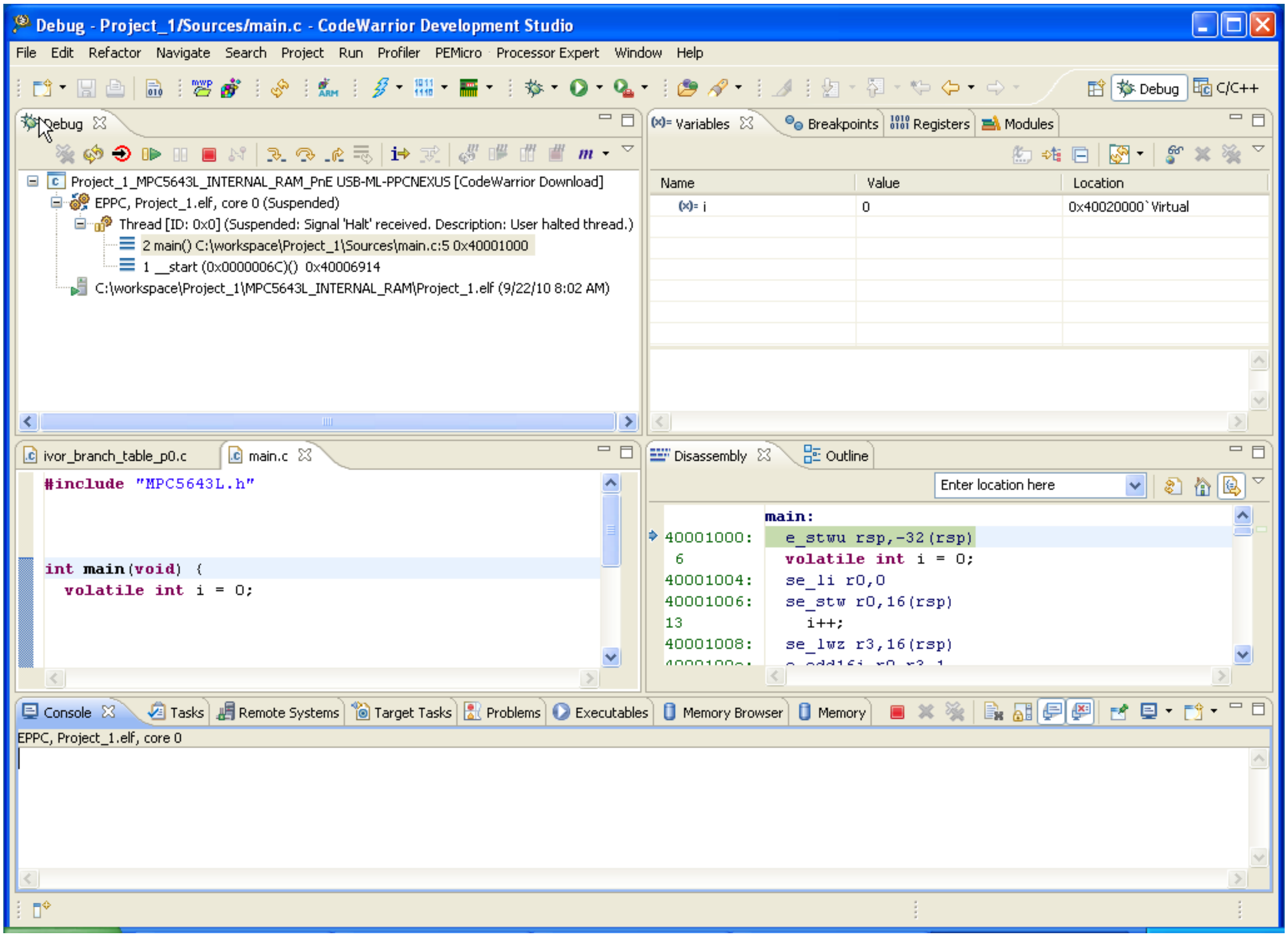
Figure 6-5. Multicore Debugging - Debug All Cores

NOTE

When you enter debug, the DPM project will show you two cores, first stopped at main() and the second stopped at __start_p1. For the LSM project you will see only one core stopped at main().

7. Select a thread from core 0 in the **Debug** view.

   All the views in the **Debug** perspective will be updated to display the debug session for the selected core.



```
Project_1_MPC5643L_INTERNAL_RAM_PnE USB-ML-PPCNEXUS [CodeWarrior Download]
  EPPC, Project_1.elf, core 0 (Suspended)
    Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
      2 main() C:\workspace\Project_1\Sources\main.c:5 0x40001000
      1 __start (0x0000006C)() 0x40006914
  C:\workspace\Project_1\MPC5643L_INTERNAL_RAM\Project_1.elf (9/22/10 8:02 AM)
```

Name	Value	Location
(x)= i	0	0x40020000`Virtual

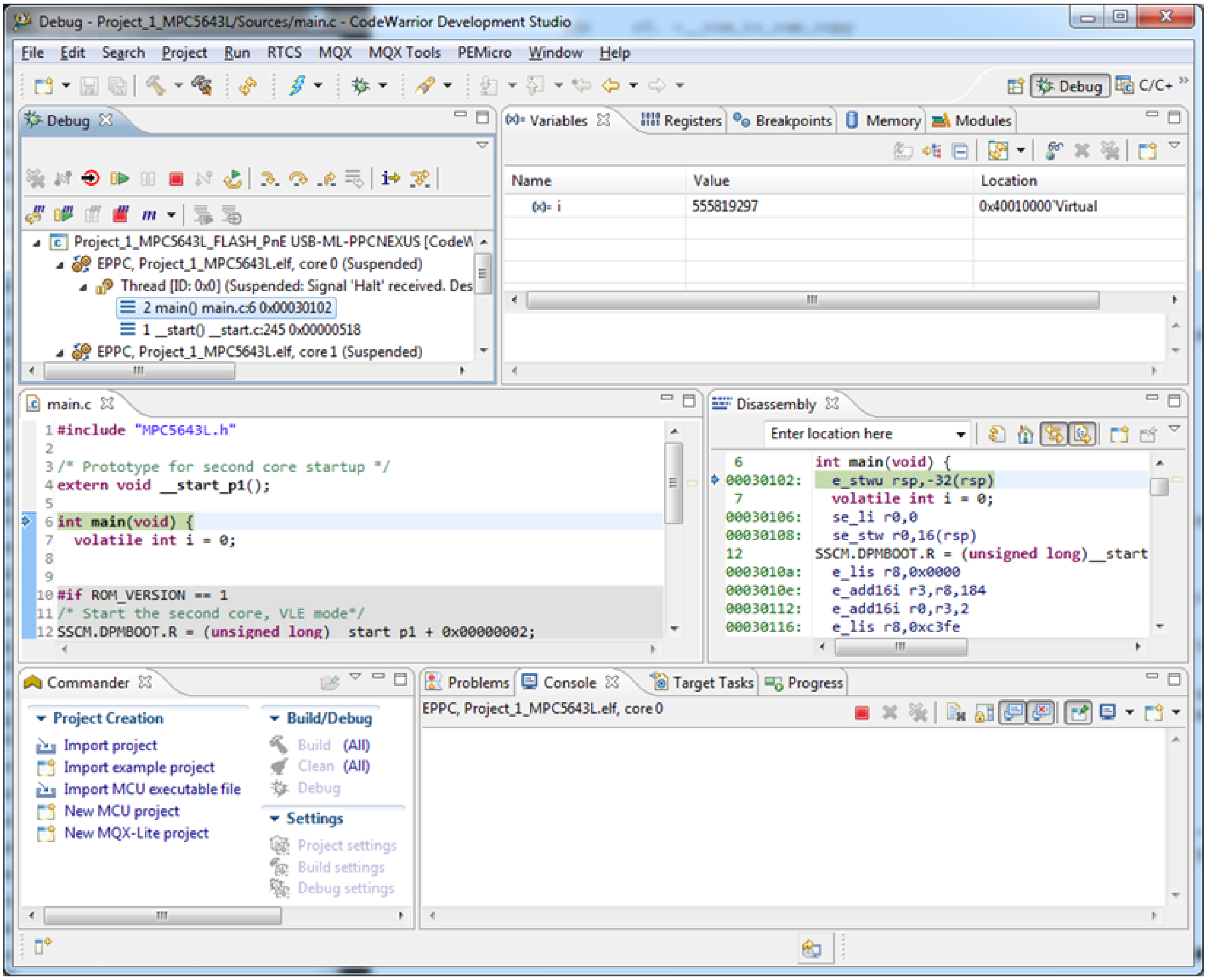
```
#include "MPC5643L.h"


int main(void) {
  volatile int i = 0;
```

```
                main:
40001000:         e_stwu rsp,-32(rsp)
 6                volatile int i = 0;
40001004:         se_li r0,0
40001006:         se_stw r0,16(rsp)
13                  i++;
40001008:         se_lwz r3,16(rsp)
```

```
EPPC, Project_1.elf, core 0
```

Figure 6-6. Debugging a LSM Project


Debug - Project_1_MPC5643L/Sources/main.c - CodeWarrior Development Studio
File Edit Search Project Run RTCS MQX MQX Tools PEMicro Window Help
Debug C/C++
Debug
Variables Registers Breakpoints Memory Modules
main.c
Disassembly
Enter location here
Commander
Project Creation
Import project
Import example project
Import MCU executable file
New MCU project
New MQX-Lite project
Build/Debug
Build (All)
Clean (All)
Debug
Settings
Project settings
Build settings
Debug settings
Problems Console Target Tasks Progress

```
Project_1_MPC5643L_FLASH_PnE USB-ML-PPCNEXUS [CodeW
  EPPC, Project_1_MPC5643L.elf, core 0 (Suspended)
    Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Des
      2 main() main.c:6 0x00030102
      1 __start() __start.c:245 0x00000518
  EPPC, Project_1_MPC5643L.elf, core 1 (Suspended)
```

Name	Value	Location
(x)= i	555819297	0x40010000 Virtual

```
#include "MPC5643L.h"

/* Prototype for second core startup */
extern void __start_p1();

int main(void) {
  volatile int i = 0;


#if ROM_VERSION == 1
/* Start the second core, VLE mode*/
SSCM.DPMBOOT.R = (unsigned long)  start p1 + 0x00000002;
```

```
 6          int main(void) {
00030102:     e_stwu rsp,-32(rsp)
 7            volatile int i = 0;
00030106:     se_li r0,0
00030108:     se_stw r0,16(rsp)
12          SSCM.DPMBOOT.R = (unsigned long)__start
0003010a:     e_lis r8,0x0000
0003010e:     e_add16i r3,r8,184
00030112:     e_add16i r0,r3,2
00030116:     e_lis r8,0xc3fe
```

```
EPPC, Project_1_MPC5643L.elf, core 0
```

Figure 6-7. Viewing Debug Information for Core 0

6.3 Debugging Multicore Project

The debugger in CodeWarrior Development Suite provides the facility to debug multiple processors using a single debug environment. The run control operations can be operated independently or synchronously. A common debug kernel facilitates multicore, run control debug operations for examining and debugging the interaction of the software running on the different cores on the system. This topic describes how the CodeWarrior debugger connects and downloads software to separate cores in a multicore processor and how you can choose to debug a program on a specific core.

NOTE

This procedure assumes that you have already created a Multicore project named `Project_1`.

To debug multicore projects, perform these steps.

- Targeting Core
- Starting Debugging Session for Core
- Debugging Specific Core
- Multicore Operations

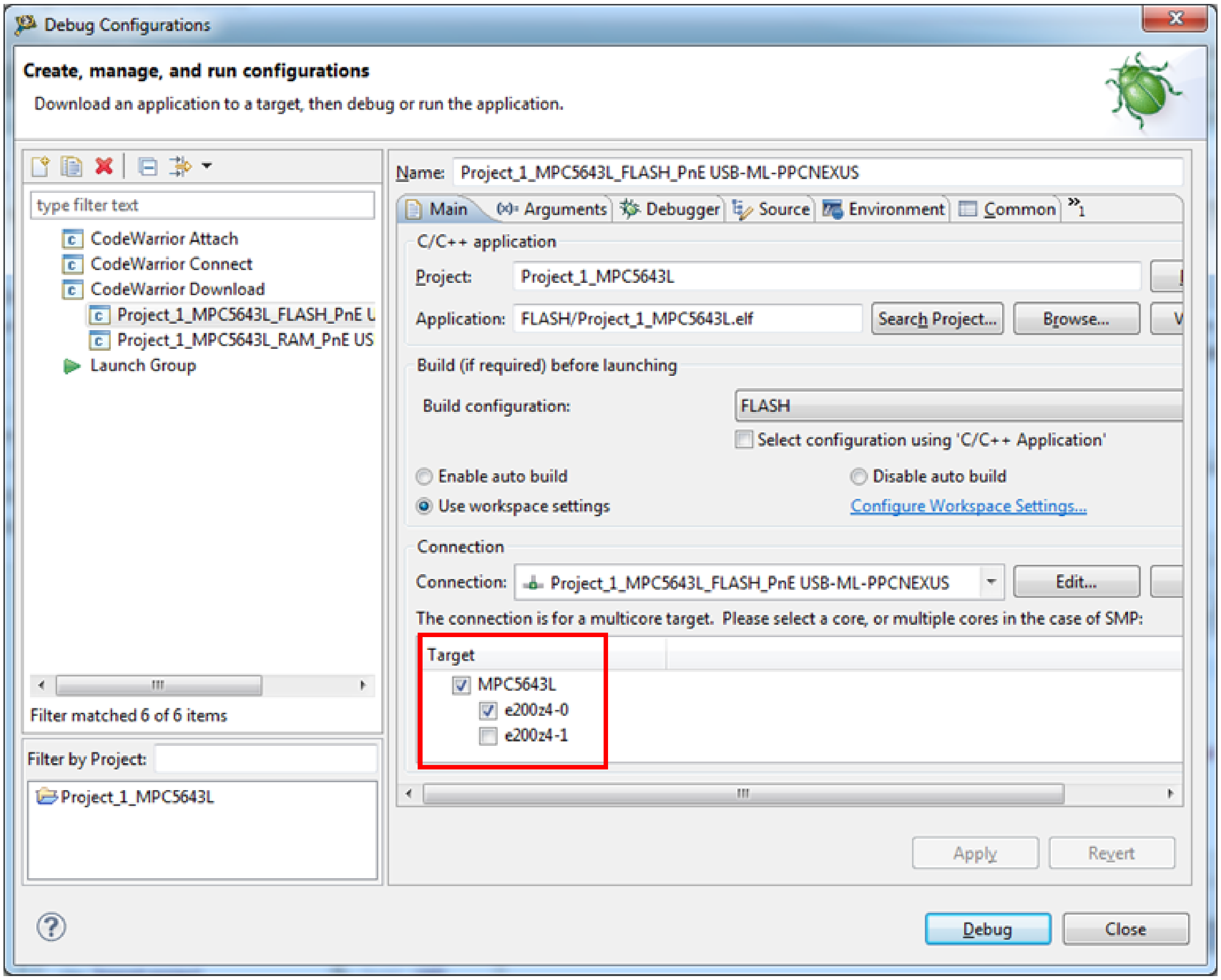
6.3.1 Targeting Core

The CodeWarrior debugger connects to specific processor core through information stored in a Download launch configuration. Specifically, the core index value on the **Main** tab of the **DebugConfigurations** dialog box determines the core targeted for debug operations.

NOTE

The core index value starts from zero. That is, the first processor core has an index value of zero, the second processor core has an index of 1, and so on.

You can set this core index value on the **Main** tab on the **Debug Configurations** dialog box, when you are modifying the settings of a Download launch configuration.



Name: Project_1_MPC5643L_FLASH_PnE USB-ML-PPCNEXUS

C/C++ application

Project: Project_1_MPC5643L

Application: FLASH/Project_1_MPC5643L.elf

Build (if required) before launching

Build configuration: FLASH

☐ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Connection

Connection: Project_1_MPC5643L_FLASH_PnE USB-ML-PPCNEXUS

The connection is for a multicore target. Please select a core, or multiple cores in the case of SMP:

Target

☑ MPC5643L

☑ e200z4-0

☐ e200z4-1

Figure 6-8. Debug Configurations Dialog Box

6.3.2 Starting Debugging Session for Core

To connect the debugger to a specific core and start a debugging session, start the Download launch configuration for that core. You start a Download launch configuration from either the **Run** menu or the toolbar's **Debug** icon.

- From Debug Configurations Dialog Box
- From Run Menu
- From Toolbar's Debug Icon

6.3.2.1 From Debug Configurations Dialog Box

1. Select **Run > Debug Configurations** .

   The **Debug Configurations** dialog box appears.

2. Pick the desired launch configuration from the left-hand side panel.
3. In the **Main** tab, select the core you want to debug.

6.3.2.2 From Run Menu

1. Choose **Run > Debug History** .

   A menu of launch configurations appears.

2. Select the desired launch configuration from this menu.
3. In the **Main** tab, select the core you want to debug.

6.3.2.3 From Toolbar's Debug Icon

1. Click the **Debug** icon in the IDE's toolbar in the **Debug** perspective.

   A menu of launch configurations appears.

2. Select the desired launch configuration from this menu.
3. In the **Main** tab, select the core you want to debug.

6.3.3 Debugging Specific Core

After you click **Debug** select the launch configuration, the debugger downloads the program to the specific core and the **Debug** perspective appears. Within the **Debug** view, the program’s thread appears. The thread is identified by its launch configuration name and the index value of the core that it executes on. If you are source code debugging, the program’s source appears in an editor view.

To debug a specific core's program, click on its thread in the **Debug** view. The **Debug** perspective automatically displays the source, registers, and variables for this core. If you click on another thread, the **Debug** perspective updates all of the views to display that core's context.

6.3.4 Multicore Operations

This topic explains the various features available to you when debugging a multicore processor.

- Multicore Commands in CodeWarrior IDE
- Multicore Commands in Debugger Shell

6.3.4.1 Multicore Commands in CodeWarrior IDE

When you start a multicore debug session, multicore commands are enabled on the CodeWarrior IDE **Run** menu. These commands, when issued, affect all cores simultaneously. The table below describes each menu choice. For detailed information on these commands, refer the **CodeWarrior Common Features Guide** .

Table 6-1. Multicore Debugging Commands

Command	Icon	Description
Multicore Resume	▶	Starts all cores of a Multicore system running simultaneously.
Multicore Suspend	⏸	Stops execution of all cores of a Multicore system simultaneously.
Multicore Restart	↻	Restarts all the debug sessions for all cores of a Multicore system simultaneously.
Multicore Terminate	■	Kills all the debug sessions for all cores of a Multicore system simultaneously.

NOTE

The **Resume** and **Step Into** buttons are enabled whenever one or more selected cores are in the stopped state. The **Suspend** button is enabled whenever one or more selected cores are in the running state. The **Restart** and **Terminate** buttons are enabled whenever one or more cores are being debugged.

To use the Multicore commands from the **Debug** perspective, perform these steps.

1. Start a debugging session by selecting the appropriately configured launch configuration.
2. If necessary, expand the desired core's list of active threads by clicking on the tree control.
3. Click on the thread you want to use with multicore operations.

NOTE

Selecting a thread uses the Multicore Group the core is part of in the multicore. For more information on the Multicore Groups feature, refer to the CodeWarrior Common Features Guide.

4. From the **Run** menu, specify the multicore operation to perform on the thread.

NOTE

The keyboard shortcut for the **Multicore Resume** operation is Shift-F8.

6.3.4.2 Multicore Commands in Debugger Shell

In addition to the multicore-specific toolbar buttons and menu commands available in the **Debug** view, the **Debugger Shell** has Multicore specific commands that can control the operation of one or more processor cores at the same time. Like the menu commands, the Multicore debugger shell commands allow you to select, start, and stop a specific core. You can also restart or kill sessions executing on a particular core. The table below lists and defines the affect of each Multicore debugging command.

Table 6-2. Multicore Debugging Commands

Command	Shortcut	Description

`mc::go`	`mc::g`	Resume multiple cores
		Syntax
		`go`
		Examples
		`mc::go`
		Resumes the selected cores associated with the current thread context.

`mc::kill`	`mc::kill`	Terminates the debug session for selected cores associated with the current thread context. **Syntax** `mc::kill` **Examples** `mc::kill` Terminates multiple cores.

Table continues on the next page...

Table 6-2. Multicore Debugging Commands (continued)

Command	Shortcut	Description
`mc::restart`	`mc::restart`	Restarts the debug session for selected cores associated with the current thread context. **Syntax** `mc::restart` **Examples** `mc::restart` Restarts multiple cores
`mc::stop`	`mc::stop`	Stops the selected cores associated with the current thread context. **Syntax** `mc::stop` **Examples** `mc::stop` Suspends multiple cores

`mc::group`	`mc::gr`	Display or edit multicore groups
		Syntax
		`group group new <type-name> [<name>] group rename <name>|<group-index> <new-name>group remove <name>|<group-index> ... group removeall group enable|disable <index> ...|all`
		Examples
		`mc::group`
		Shows the defined groups, including indices for use in the `mc::group rename|enable|remove` set of commands.
		`mc::group new 8572`
		Creates a new group for system type `8572`. The group name will be based on the system name and will be unique. The enablement of the group elements will be all non-cores enabled, all cores disabled.
		`mc::group rename 0 "My Group Name"`
		Renames the group at index 0 to "My Group Name".
		`mc::group enable 0 0.0`

Table continues on the next page...

Table 6-2. Multicore Debugging Commands (continued)

Command	Shortcut	Description

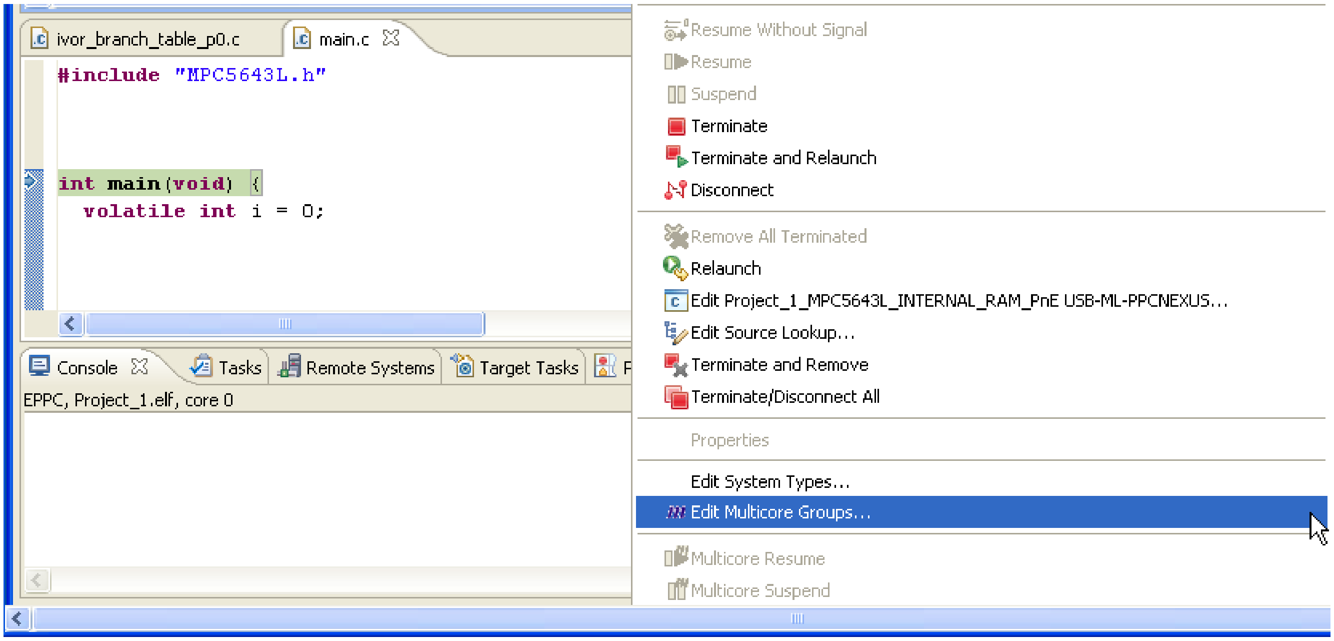
		Enables the group at index 0 and the element at index 0.0 of the mc::group command.
		`mc::group remove "My Group Name"`
		Removes the group named "My Group Name".
		`mc::group removeall`
		Removes all groups.

mc::type	mc::t	Shows the system types available for multicore debugging as well as type indices for use by the `mc::type remove` and `mc::group new` commands.
		Syntax
		`type type import <filename> type remove <filename>|<type-index> ... type removeall`
		Examples
		`mc::type`
		Display or edit system types
		`mc::type import 8572_jtag.txt`
		Creates a new type from the JTAG configuration file.
		`mc::type remove 8572_jtag.txt`
		Removes the type imported from the specified file.
		`mc::group removeall`
		Removes all imported types.

6.4 Editing Multicore Groups

To edit a multicore group, perform these steps.

1. Right-click on the launch configuration in the **Debug** perspective and select **Edit Multicore Group** from the context menu.



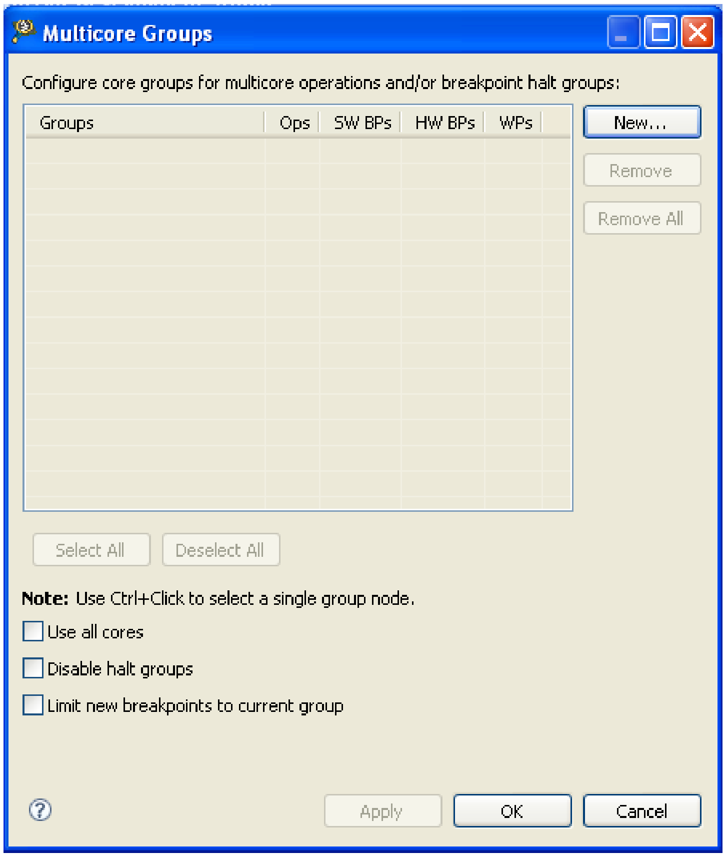
```
#include "MPC5643L.h"


int main(void) {
  volatile int i = 0;
```

EPPC, Project_1.elf, core 0

Figure 6-9. Edit Multicore Group in Context Menu

The **Multicore Groups** dialog box appears.



Configure core groups for multicore operations and/or breakpoint halt groups:

Groups	Ops	SW BPs	HW BPs	WPs

Note: Use Ctrl+Click to select a single group node.

□ Use all cores

□ Disable halt groups

□ Limit new breakpoints to current group

Figure 6-10. Multicore Groups Dialog Box

2. Click **New**.

The **New Multicore Group** dialog box appears. This is a simple selection dialog consisting of all types of multicore systems known to the product as well as any imported multiprocessor systems.

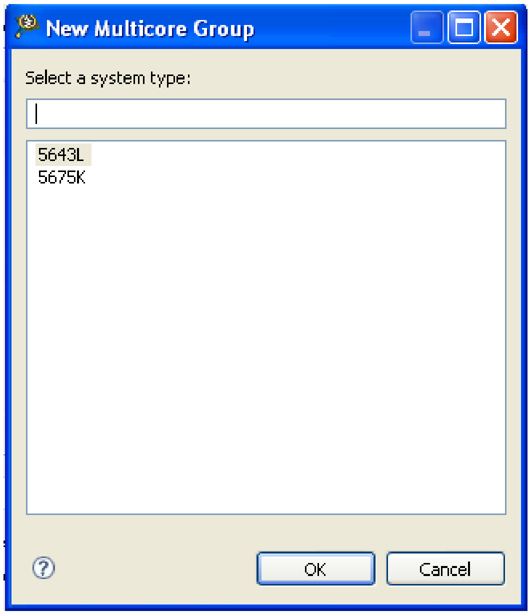
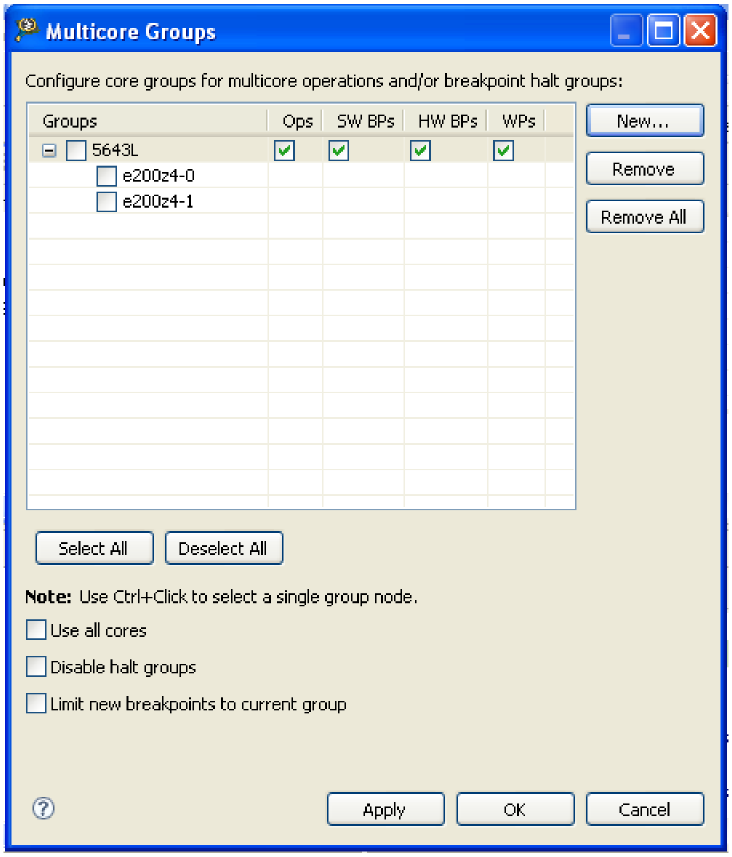


Figure 6-11. New Multicore Group Dialog Box

3. Select a system type and click **OK** .

   The selected system type appears in the **Multicore Groups** dialog box.



Configure core groups for multicore operations and/or breakpoint halt groups:

Groups	Ops	SW BPs	HW BPs	WPs
⊟ □ 5643L	☑	☑	☑	☑
□ e200z4-0				
□ e200z4-1				

Note: Use Ctrl+Click to select a single group node.

□ Use all cores

□ Disable halt groups

□ Limit new breakpoints to current group

Figure 6-12. Multicore Groups Dialog Box

The tree view in the dialog shows the list of defined groups. The groups are the top-level nodes in the tree, and each has an editable checkbox to specify whether the group is enabled or not. Each group contains one or more children, each representing a processor or a core. A processor will, in turn, have children representing the cores in the processor. Each processor/core node has an editable checkbox for choosing whether that element will be included in the group (root node). The buttons are used as follows:

- **New** - Click to create a new group via the New Multicore Group dialog. The initial name of the group will be the system type name unless the name is already in use, in which case an index will be appended as is customary, e.g., '8572 (1)'. The initial enablement of the group and its descendants will be non-cores enabled, cores disabled. This guarantees an initial state with no error due to overlap.
- **Remove** - Click to remove the selected group. The button is disabled if no groups exist in the list or if no group is selected.
- **Remove All** - Click to remove all groups. The button is disabled if no groups exist in the list.

- **Select All** - Click to enable all groups, processors and cores. The button is disabled if no groups exist in the list.
- **Deselect All** - Click to disable all groups, processors and cores. The button is disabled if no groups exist in the list.

NOTE

The buttons **Select All** and **Deselect All** are used as is customary in other dialog boxes to mean **Enable/ Disable All Checkboxes** .

4. Click **Apply** .
5. Click **OK** .

6.5 Editing Target Types

To edit a system type, perform these steps.

1. Right-click on the launch configuration in the **Debug** perspective and select **Edit Target Types** from the context menu.

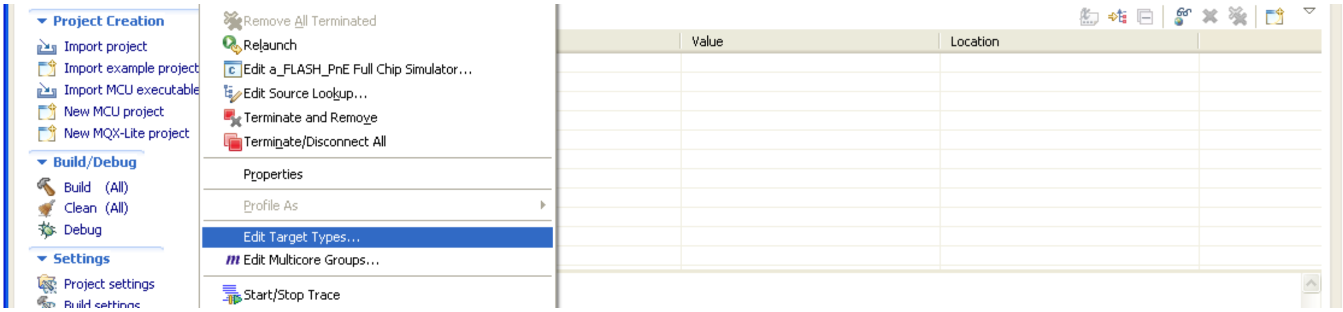


Figure 6-13. Edit Target Types in Context Menu

The **Target Types** dialog box appears.

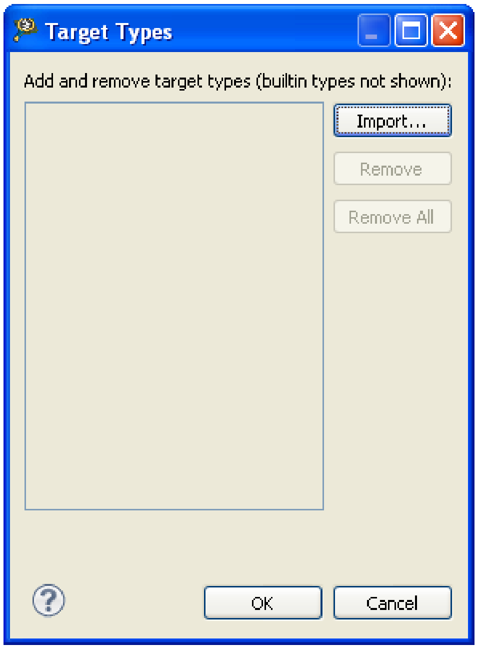


Figure 6-14. System Types Dialog Box

- **Import** - Creates a custom target type by importing it from a configuration file.
- **Remove** - Removes a target type from the list.
- **Remove All** - Removes all target types from the list.

2. Click **Import**.

   The **Import Target Type** dialog box appears.

3. Select the desired file and click **Open**.

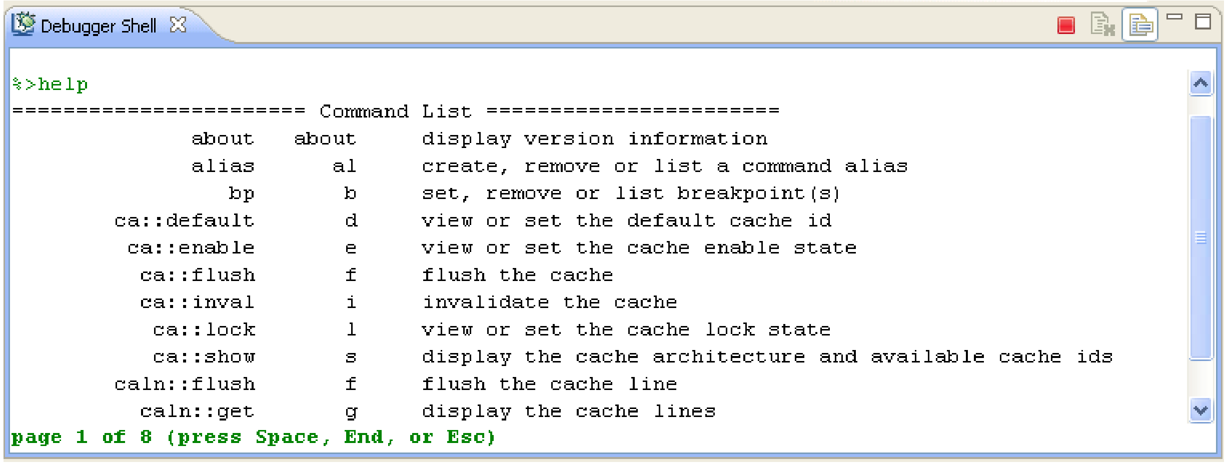
   The selected file appears in the **Import Target Type** dialog box.

   The **Import Target Type** dialog box lists the current imports, with buttons to remove existing entries and to import new entries. If you attempt to remove a type that has associated groups, you will be asked whether you want to continue, in which case all associated groups will be removed as well. Similarly, if you try to remove a type that has associated launch configurations, you will be asked whether you want to continue, in which case all associated launch configurations will be reset to the uniprocessor type.

Chapter 7
CodeWarrior Command Line Debugging

CodeWarrior supports a command-line interface to some of its features including the debugger. You can use the command-line interface together with various scripting engines, such as the Microsoft® Visual Basic® script engine, the Java™ script engine, Tcl, Python, and Perl. You can even issue a command that saves the command-line activity to a log file.

You can use the **Debugger Shell** view to issue command lines to the IDE. For example, you can enter the command `debug` in this window to start a debugging session. The window lists the standard output and standard error streams of command-line activity.



```
%>help
======================= Command List =======================
              about   about     display version information
              alias      al     create, remove or list a command alias
                 bp       b     set, remove or list breakpoint(s)
        ca::default       d     view or set the default cache id
         ca::enable       e     view or set the cache enable state
          ca::flush       f     flush the cache
          ca::inval       i     invalidate the cache
           ca::lock       l     view or set the cache lock state
           ca::show       s     display the cache architecture and available cache ids
        caln::flush       f     flush the cache line
          caln::get       g     display the cache lines
page 1 of 8 (press Space, End, or Esc)
```

Figure 7-1. Debugger Shell View

To open the **Debugger Shell** view, perform these steps.

1. Switch the IDE to the **Debug** perspective and start a debugging session.
2. Select **Window > Show View > Debugger Shell**.

   The **Debugger Shell** view appears.

NOTE

Alternatively, select **Window > Show View > Other** . Expand the **Debug** tree control in the **Show View** dialog box, select **Debugger Shell**, and click **OK**.

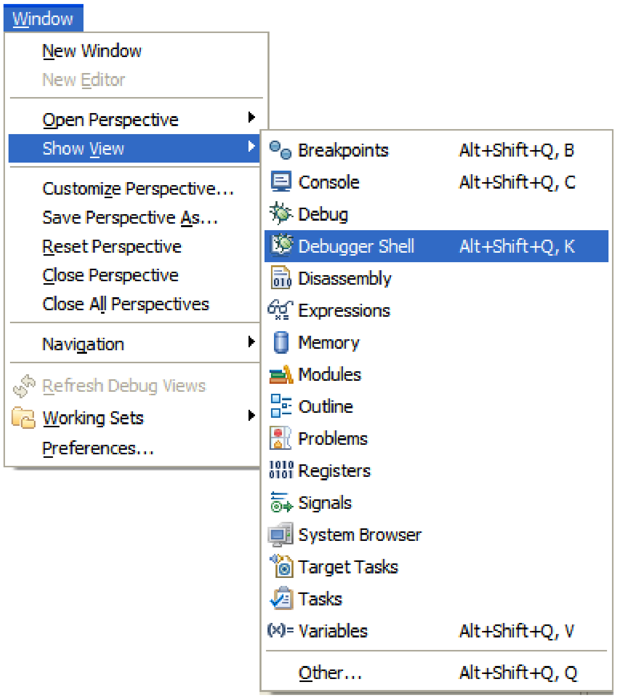


Figure 7-2. Show View - Debugger Shell

To issue a command-line command, type the desired command at the command prompt (`%>`) in the **Debugger Shell** view, then press Enter or Return. The command-line debugger executes the specified command.

If you work with hardware as part of your project, you can use the command-line debugger to issue commands to the debugger while the hardware is running.

NOTE

To list the commands the command-line debugger supports, type `help` at the command prompt and press Enter. The `help` command lists each supported command along with a brief description of each command.

Tip

To view page-wise listing of the debugger shell commands, right-click in the **Debugger Shell** view and select **Paging** from the context menu. Alternatively, click the **Enable Paging** icon.

When you debug from the command line, you can use:

- Tcl Support

- Command-Line Debugging Tasks
- Debugger Shell Command List
- Microcontrollers-Specific HIWARE Commands

7.1 Tcl Support

This topic describes command-line debugger's Tcl support.

7.1.1 Resolution of Conflicting Command Names

The names of several command-line debugger commands conflict with the Tcl commands. The table below explains how the command-line debugger resolves such conflicts (if the mode is set to auto).

Table 7-1. Resolving Conflicting Commands

Command	Resolution
load	If you pass the command-line debugger a load command that includes a filename containing the suffix .eld or .mcp, the debugger loads the project. Otherwise, the debugger invokes the Tcl load command.
bp	If you pass the command-line debugger a `bp` command from within a script and the command has no arguments, the debugger invokes the `Tcl break` command. Otherwise, the debugger interprets a `break` command as a command to control breakpoints.
close	If you pass the command-line debugger a close command that has no arguments, the debugger terminates the debug session. Otherwise, the debugger invokes the Tcl close command.

7.1.2 Execution of Script Files

Tcl usually executes a script file as one large block, returning only after execution of the entire file. For the `run` command, however, the command-line debugger executes script files line-by-line. If a particular line is not a complete `Tcl` command, the debugger appends the next line. The debugger continues appending lines until it gets a complete Tcl script block.

The listing below lists code that includes a script. For the `Tclsource` command, the debugger executes this script as one block. But for the run debug command, the debugger executes this script as two blocks, the set statement and the while loop.

Listing: Example Tcl Script

```
set x 0;
while {$x < 5}

{

puts "x is $x";

set x [expr $x + 1]

}
```

NOTE

> The `run` debug command synchronizes debug events between blocks in a script file. For example, after a `go`, `next`, or `step` command, `run` polls the debug thread state and does not execute the next line or block until the debug thread terminates. However, the Tcl source command does not consider the debug thread state. Consequently, use the `run` debug command to execute script files that contain these debug commands: `debug`, `go`, `next`, `stop`, and `kill`.

7.1.3 Tcl Startup Script

The command-line debugger can automatically run a Tcl script each time you open the command-line debugger window. This script is called a startup script.

You can use both Tcl and command-line debugger commands in the startup script. For example, you might include commands that set an alias or a define color configuration in a startup script.

To create a command-line debugger startup script, follow these steps.

1. Put the desired Tcl and command-line debugger commands in a text file.

2. Name this file `tcld.tcl`.
3. Place `tcld.tcl` in one of the directories listed below.
   - On a Windows® PC, put `tcld.tcl` in the system directory.

     For example, on Windows XP, put `tcld.tcl` in the WINDOWS directory.

   - On a Solaris Workstation, put `tcld.tcl` in your home directory.

NOTE

There is no synchronization of debug events in the startup script. Consequently, put the `c` debug command to the startup script and place the debug commands `debug`, `go`, `stop`, `kill`, `next`, and `step` in another script so that they execute successfully.

7.1.4 Command-Line Syntax

Start the CodeWarrior Eclipse IDE and execute a Debugger Shell script with a TclScript script as input from the command-line, as shown in the example below:

```
D:\MCU\eclipse>cwide.exe -vmargsplus -Dcw.script=D:\my_script.tcl
```

NOTE

Users familiar with the `-vmargs` option in the CodeWarrior Eclipse IDE should note that CodeWarrior will not work properly if `-vmargs` is used. Use the custom `-vmargsplus` option in place of the `-vmargs` option.

7.2 Command-Line Debugging Tasks

The table below provides instructions for common command-line debugging tasks.

Table 7-2. Common Command-Line Debugging Tasks

Task	Instruction	Comments
Open the Debugger Shell	Select **Windows > Show View > Others > Debugger Shell**	The **Debugger Shell** view appears.
Use the help command	1. On the Debugger shell command prompt (**%>**), type help. 2. Press Enter.	The Command List for CodeWarrior is appears.

Table continues on the next page...

Table 7-2. Common Command-Line Debugging Tasks (continued)

Task	Instruction	Comments
Enter a command	1. On the Debugger shell, type a command followed by a space. 2. Specify any valid command-line options, separating each with a space. 3. Press Enter.	You can use shortcuts instead of complete command names, such as k for kill.
View debug command hints	Specify alias followed by a space	The syntax for the rest of the command appears.
Review previous commands	Press Up Arrow and Down Arrow keys	
Clear command from the command line	Press the Esc key	
Stop an executing script	Press the Esc key	
Toggle between insert/overwrite mode	Press the Insert key	
Scroll up/ down a page	Press Page Up or Page Down key	
Scroll left/right one column	Press Ctrl-Left Arrow or Ctrl-Right Arrow keys	
Scroll to beginning or end of buffer	Press Ctrl-Home or Ctrl-End keys	

7.3 Debugger Shell Command List

This topic lists and defines each command-line debugger command.

NOTE

This chapter contains only the commands specific to the architectures supported in MCU product. The complete list of commands is available in *CodeWarrior Common Features Guide*.

7.3.1 cmdwin::eppc::getcoreid

Get the current core ID.

Syntax

```
cmdwin::eppc::getcoreid
```

Examples

```
cmdwin::eppc::getcoreid
```

Display the current core ID value.

7.3.2 cmdwin::eppc::setMMRBaseAddr

Set the MMR base address.

Syntax

```
cmdwin::eppc::SetMMRBaseAddr  <addr>
```

Examples

```
cmdwin::eppccmdwin::eppc::SetMMRBaseAddr 0x04700000
```

Tell the debugger to use 0x04700000 and the MMR base address.

All memory mapped register reads and writes will use this new base address.

7.3.3 cmdwin::eppc::setcoreid

Set the core ID.

Syntax

```
cmdwin::eppc::setcoreid  <coreID>
```

Examples

The table below lists and defines the examples of the `cmdwin::eppc::setcoreid` command.

Table 7-3. cmdwin::eppc::setcoreid Command-Line Debugger Command - Examples

Command	Description
`cmdwin::eppc::setcoreid 1`	Set the core ID value to 1.
`cmdwin::eppc::setcoreid default`	Set the core ID value to default. All commands will be executed on the new set core. To see the current core ID use `getcoreid` command.

7.3.4 gdi

Forwards third party custom commands.

Syntax

```
gdi<custom cmd> [<custom cmd arg list>]
```

Examples

```
gdi help
```

Forwards help command to currently selected GDI connection. The purpose of this command is to forward third party custom commands to their debug instrument, which is currently opened.

7.4 Microcontrollers-Specific HIWARE Commands

This topic lists and defines Microcontrollers-specific HIWARE commands.

7.4.1 Command List

The table below lists the supported HIWARE commands followed by:

- CodeWarrior debugger shell syntax,
- partially supported commands,
- commands that are not applicable in CodeWarrior
- commands supported in script files with Tcl control flow statements, and
- unsupported commands.

The following columns represent the status in the CodeWarrior Eclipse IDE:

- *S-CW* - Supported command which follows CodeWarrior debugger shell syntax
- *P* - Command is partially supported, meaning some options/parameters are not supported
- *NA* - Command is not applicable in CodeWarrior
- *S-Tcl* - Commands is supported in script files with Tcl control flow statements
- *U* - Command is not supported

Table 7-4. Microcontrollers-Specific Debugger Command List

Command	Status	Description
HIWARE		
VER	*S-CW*	Lists the version of all loaded commands

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
		Syntax `about`
AUTOSIZE	*NA*	Selects window sizing mode
OPENIO	*NA*	Loads an IO simulation component
OPENPROJECT	*U*	Opens an existing project
OPEN	*NA*	Opens a component window
SET	*U*	Loads a target component
LOAD	*U*	Loads an application (Code & Symbols)
LOADCODE	*U*	Loads an application (Code only)
LOADSYMBOLS	*S-CW*	Loads an application (Symbols only) **Syntax** `loadsym <filename>`
FONT	*U*	Changes font in component windows
BCKCOLOR	*U*	Changes background color of component windows
SLAY	*NA*	Saves the layout and options of all components
ACTIVATE	*NA*	Activates a window component (in/out focus)
CLOSE	*NA*	Closes a component window
SYSTEM	*S-CW*	Executes an external application **Syntax** `system <command>`
EXIT	*S-CW*	Terminates this application **Syntax** `quitIDE`
RESET	*S-CW*	Resets the target MCU **Syntax**`Reset`

HELP	*S-CW*	Lists available commands; to get help on a specific command, use the command followed by '?'
		Syntax
		`help`
		`help <command><command> ?`

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
HIWARE Engine		

LF	*S-CW*	Opens a log file
		Syntax (for command)
		`log c <file>`
		Syntax (for session)
		`log s <file>`

NOLF	*S-CW*	Closes a log file
		Syntax (for command)
		`log off c`
		Syntax (for session)
		`log off s`

CR	U	Records all commands to a file
NOCR	U	Stops recording commands to a file

LOG	S-CW	Specifies items to be logged
		Syntax (for command)
		`log c <file>`
		Syntax (for session)
		`log s <file>`

BS	P	Sets breakpoint
		Syntax
		`bp [-{hw|sw|auto}] {<func>|[<ms>:]<addr>|<file> <line> [<column>]}`
		`bp all|#<id>|<func>|<addr> enable|disable|{ignore <count>}`
		`bp #<id> cond <c-expr>`

SAVEBP	*U*	Saves breakpoints into a file
STEPINTO	*S-CW*	Step Into **Syntax** `step [asm|src] into`
STEPOUT	*S-CW*	Step out **Syntax** `step [asm|src] out`
STEPOVER	*S-CW*	Step over

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
		Syntax `step [asm|src] over`
RESTART	*S-CW*	Restart execution **Syntax** `restart`
DDEPROTOCOL	*U*	DDE Protocol options
DEFINEVALUEDLG	*U*	Opens a GUI to define a value for the symbol/variable given as parameter
BC	*S-CW*	Clears breakpoint **Syntax** `bp all|#<id>|<func>|<addr> off`
BD	*S-CW*	Lists breakpoints **Syntax** `bp`
GO	S-CW	Starts execution (Go) **Syntax** `go`
STOP	S-CW	Stops execution (Halt) **Syntax** `stop`
P	S-CW	Executes an instruction (Flat step) **Syntax** `stepi`
T	S-CW	Executes CPU instructions **Syntax** `stepi`
Configuration Example • `radix x` • `config MemIdentifier 0` • `config MemWidth 32` • `config MemAccess 32` • `config MemSwap off` **Note** : These options apply only to the memory commands below.		
WB	S-CW	Writes byte(s) into target memory

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
		Syntax `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] =<value>`
WW	S-CW	Writes word(s) into target memory (2 bytes) **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] =<value>`
WL	S-CW	Writes long(s) into target memory (4 bytes) **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] =<value>`
MS	S-CW	Writes byte(s) into target memory (same as WB) **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] =<value>`
RD	S-CW	Lists registers **Syntax** `reg all`
RS	S-CW	Sets registers **Syntax** `reg <reg-spec>{..<reg>|#<n>} [-s|-ns] [%<conv>] =<value>`
MEM	U	Lists memory map
DASM	S-CW	Disassembles target memory **Syntax** `disassemble pc|[<ms>:]<addr> [<count>]`
DB	S-CW	Lists byte(s) from target memory **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] [-np]`

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
DW	S-CW	Lists words from target memory (2 bytes) **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] [-np]`
DL	S-CW	Lists long(s) from target memory (4 bytes **)** **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] [-np]`
CD	S-CW	Lists or changes directory **Syntax** `cd`
E	S-CW	Evaluates an expression and lists its result **Syntax** `evaluate [#<format>] [-l] [<var|expr>]`

A	S-CW	Evaluates an expression and assigns its result to an existing variable
		Syntax
		`var <var-spec> [-s|-ns] [%<conv>]=[evaluate [#<format>] [-l] [<var|expr>]]`
		Example
		`var myVar = [evaluate 1+1] - assigns value "2" to "myVar"`

PRINTF	U	Display a string on the window using `printf` like format
FPRINTF	U	Write a string to a file using `fprintf` like format
NB	S-CW	Changes or displays the default number base for the value of expressions **Syntax**

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
		`evaluate [#<format>] [-l] [<var|expr>]`
LS	U	Lists also global variables and procedures of the loaded application

SREC	P	Loads of Motorola S-records from a specified file
		Syntax
		`restore -h *.lod [[<ms>:]<addr>|+<offset>] [8bit|16bit|32bit|64bit]`
		`restore -b *.lod [<ms>:]<addr> [8bit|16bit|32bit|64bit]`

SAVE	S-CW	Saves a specified block of memory to a specified file in Motorola S-record format **Syntax** `save -h|-b [<ms>:]<addr>... <filename> [-a|-o] [8bit|16bit|32bit|64bit]`
PAUSETEXT	NA	Displays a modal message box for testing purpose
TESTBOX	NA	Displays a modal message box with a given string
REGFILE	U	Loads the I/O register descriptions from a 'register file'
REGBASE	U	Sets the base address of the on-chip I/O registers
ANDB	U	Bitwise-AND with target memory byte
ANDW	U	Bitwise-AND with target memory word (2 bytes)
ANDL	U	Bitwise-AND with target memory long (4 bytes)
NANDB	U	Bitwise-NAND with target memory byte
NANDW	U	Bitwise-NAND with target memory word (2 bytes)
NANDL	U	Bitwise-NAND with target memory long (4 bytes)

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
ORB	U	Bitwise-OR with target memory byte
ORW	U	Bitwise-OR with target memory word (2 bytes)
ORL	U	Bitwise-OR with target memory long (4 bytes)
NORB	U	Bitwise-NOR with target memory byte
NORW	U	Bitwise-NOR with target memory word (2 bytes)
NORL	U	Bitwise-NOR with target memory long (4 bytes)
EXORB	U	Bitwise-EXOR with target memory byte
EXORW	U	Bitwise-EXOR with target memory word (2 bytes)
EXORL	U	Bitwise-EXOR with target memory long (4 bytes)
MEMCOPY	S-CW	Copies the target memory
MEMBITCOPY	S-CW	Copies one bit from one memory address to another bit to another memory address **Syntax** `copy [<ms>:]<addr>[..<addr>|#<bytes>] [<ms>:]<addr>`
DEFINE	S-CW	Defines a symbol and associates a value **Syntax** `set varName ?value? <Tcl command>`
UNDEF	S-CW	Removes a symbol definition **Syntax** `unset varName <Tcl command>`
RETURN	U	Terminates the current command processing level
GOTO	U	Goes to the line following the label
GOTOIF	U	Goes to the line following the label if condition is TRUE
WHILE	S-Tcl	Executes commands as long as the condition is true

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
FOR	S-Tcl	Executes commands up to a predefined number of times
REPEAT	S-Tcl	Executes commands until a certain condition is true
IF	S-Tcl	Executes different command sections depending on the conditions
FOCUS	NA	Assigns a component as the destination for all subsequent commands
WAIT	S-CW	Waits by time tenths of a second **Syntax** `wait`
AT	U	Executes the next command at time (in ms)
CF	S-CW	Executes commands in the specified command file
CALL	S-CW	Executes commands in the specified command file **Syntax** `source`
Source		
SPC	NA	Highlights the statement corresponding to the code address
SMEM	NA	Highlights the statements corresponding to the code address range
SMOD	NA	Loads the corresponding module's source text
SPROC	NA	Highlights the statement of the procedure that is in the procedure chain
FOLD	NA	Hides source text for clearness at program block level
UNFOLD	NA	Exhibits the contents of folded source text blocks
SLINE	NA	Displays the line
FINDPROC	NA	Find the Procedure
FIND	NA	Searches an arbitrary pattern in the currently loaded source file
ATTRIBUTES	NA	Sets up the display
Assembly		

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
SPC	NA	Lists the specified address
SMEM	NA	Lists the specified address
ATTRIBUTES	NA	Sets up the display
Procedure		
ATTRIBUTES	NA	Sets up the display
Register		
ATTRIBUTES	NA	Sets up the display
Memory		
SPC	NA	Lists the address given as an argument
SMEM	NA	Lists the memory range given as an argument
SMOD	NA	Lists the first global variable of the module
FILL	S-CW	Fills a memory range with the given value **Syntax** `mem <addr-spec> [<range>] [-s|-ns] [%<conv>] =<value>`
UPDATERATE	NA	Sets the update rate
ATTRIBUTES	NA	Sets up the display
COPYMEM	S-CW	Copies a memory range to a specified location **Syntax** `copy [<ms>:]<addr>[..<addr>| #<bytes>] [<ms>:]<addr>`
SEARCHPATTERN	NA	Search a pattern in memory
REFRESHMEMORY	S-CW	After releasing caches, refreshes the memory **Syntax** `refresh`
Data		
SPROC	NA	Displays local or global variables of the procedure given as parameter
ADDXPR	NA	Adds a new expression in the data component
PTRARRAY	NA	Switches on or off the pointer as array displaying

Table continues on the next page...

Table 7-4. Microcontrollers-Specific Debugger Command List (continued)

Command	Status	Description
SMOD	NA	Displays global variables of the module given as parameter
ZOOM	NA	Exhibits the member fields of structures by 'diving' into the structure
UPDATERATE	NA	Sets the update rate of the data component
DUMP	P	Dumps the content of the data component to the command line **Syntax** `display`
ATTRIBUTES	NA	Sets up the display
REFRESHDATA	S-CW	After releasing caches, refreshes the display **Syntax** `refresh`
Command		
CLR	S-CW	Clears the Command window **Syntax** `cls`
ATTRIBUTES	NA	Sets up the display

Chapter 8
Build Properties for Linux Project

This chapter explains build properties for Microcontrollers Linux projects. The Microcontrollers **New Linux/ uClinux Application Project** wizard uses the information it gathers from you to set up the project's build and launch configurations.

A project's build configuration contains information on the tool settings used to make the program. For example, it describes the compiler and linker settings, and the files involved, such as source and libraries.

A project's launch configuration describes how the IDE starts the program, such as whether it executes by itself on a target, or under debugger control. Launch configurations also specify the core the program executes on (if the target processor has multiple cores).They also specify the connection interface and communications protocol that the debugger uses to control the environment that the program executes in.

NOTE

> The settings of the CodeWarrior IDE's build and launch configuration correspond to an object called a target made by the classic CodeWarrior IDE.

When the wizard completes its process, it generates launch configurations with names that follow the pattern *projectname - configtype - targettype,* where:

- *projectname* represents the name of the project
- *configtype* represents the project's name, which usually describes the build configuration
- *targettype* represents the type of target software or hardware on which the launch configuration acts

For each launch configuration, you can specify build properties, such as:

- additional libraries to use for building code
- behavior of the compilers, linkers, assemblers, and other build-related tools
- specific build properties, such as the byte ordering of the generated code

The topics in this chapter are:

- Changing Build Properties
- Restoring Build Properties
- Build Properties for Linux/uClinux Project

8.1 Changing Build Properties

The Microcontrollers **New Linux/uClinux Application Project** wizard creates a set of build properties for the project. You can modify these build properties to better suit your needs.

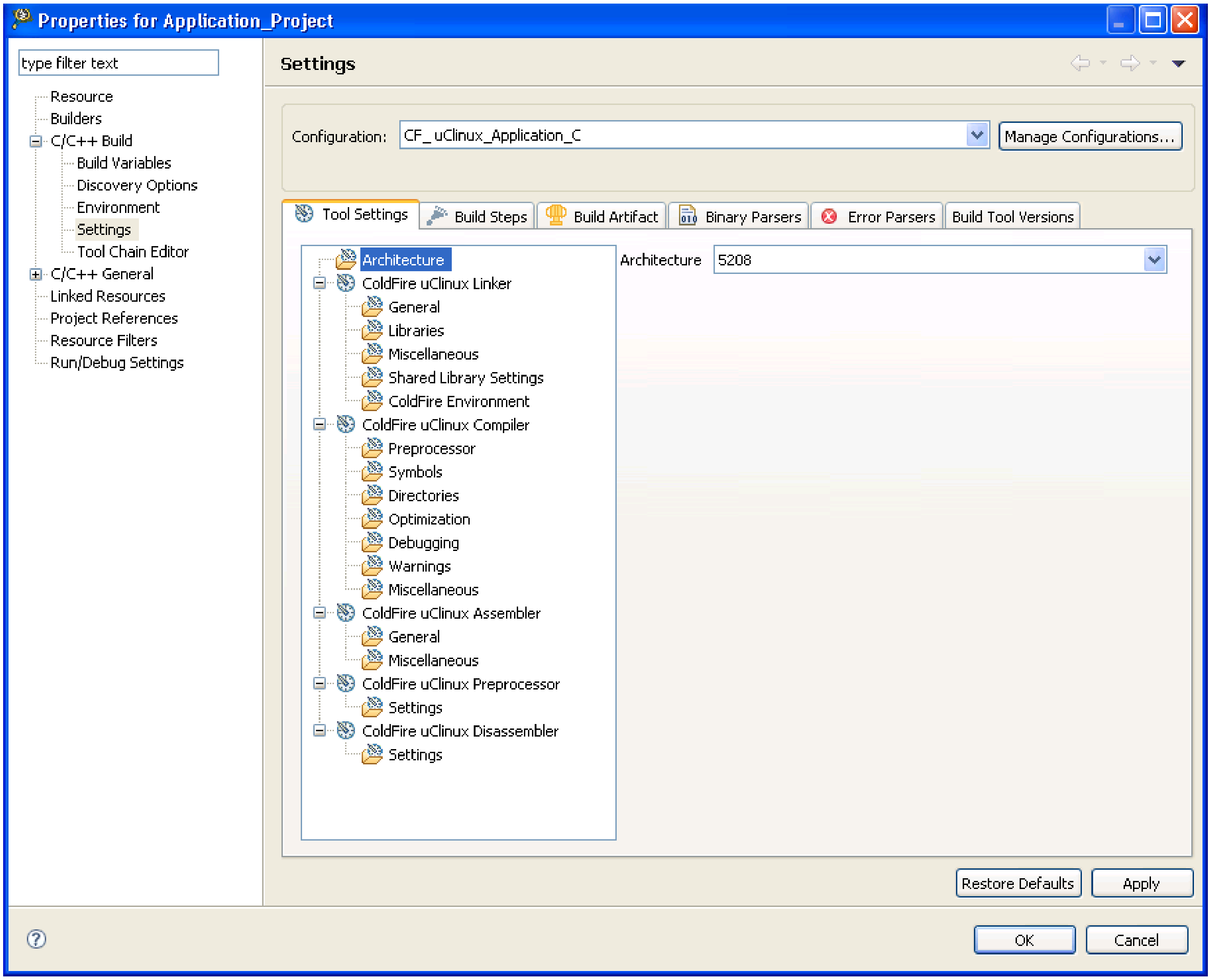
Perform these steps to change build properties:

1. Start the IDE.
2. In the **CodeWarrior Projects** view, select the project for which you want to modify the build properties.
3. Select **Project > Properties**.

   The **Properties** window appears. The left side of this window has a properties list. This list shows the build properties that apply to the current project.

4. Expand the **C/C++ Build** property.
5. Select **Settings**.

   The **Properties** window shows the corresponding build properties.



Configuration: CF_uClinux_Application_C

Architecture 5208

Figure 8-1. Properties for <*Project*> Window

6. Use the **Configuration** drop-down list to specify the launch configuration for which you want to modify the build properties.
7. Click the **Tool Settings** tab.

   The corresponding page appears.

8. From the list of tools on the **Tool Settings** page, select the tool for which you want to modify properties.
9. Change the settings that appear in the page.
10. Click **Apply**.

    The IDE saves your new settings.

You can select other tool pages and modify their settings. When you finish, click **OK** to save your changes and close the **Properties** window.

NOTE

To build any Linux/uClinux application debug projects, ensure that the CodeSourcery GCCs are located on your machine. Usually they are located in the **Cross_Tools** folder in the CodeWarrior installation. Also, you must ensure that you have two environment variables defined for the two GCCs: Embedded Linux and uClinux. `CFGCCUCINSTALLDIR` referring to the `bin` folder of the `CodeSourcery uClinux GCC` install directory. `CFGCCINSTALLDIR` referring to the `bin` folder of the `CodeSourcery GNU Linux GCC install` directory.

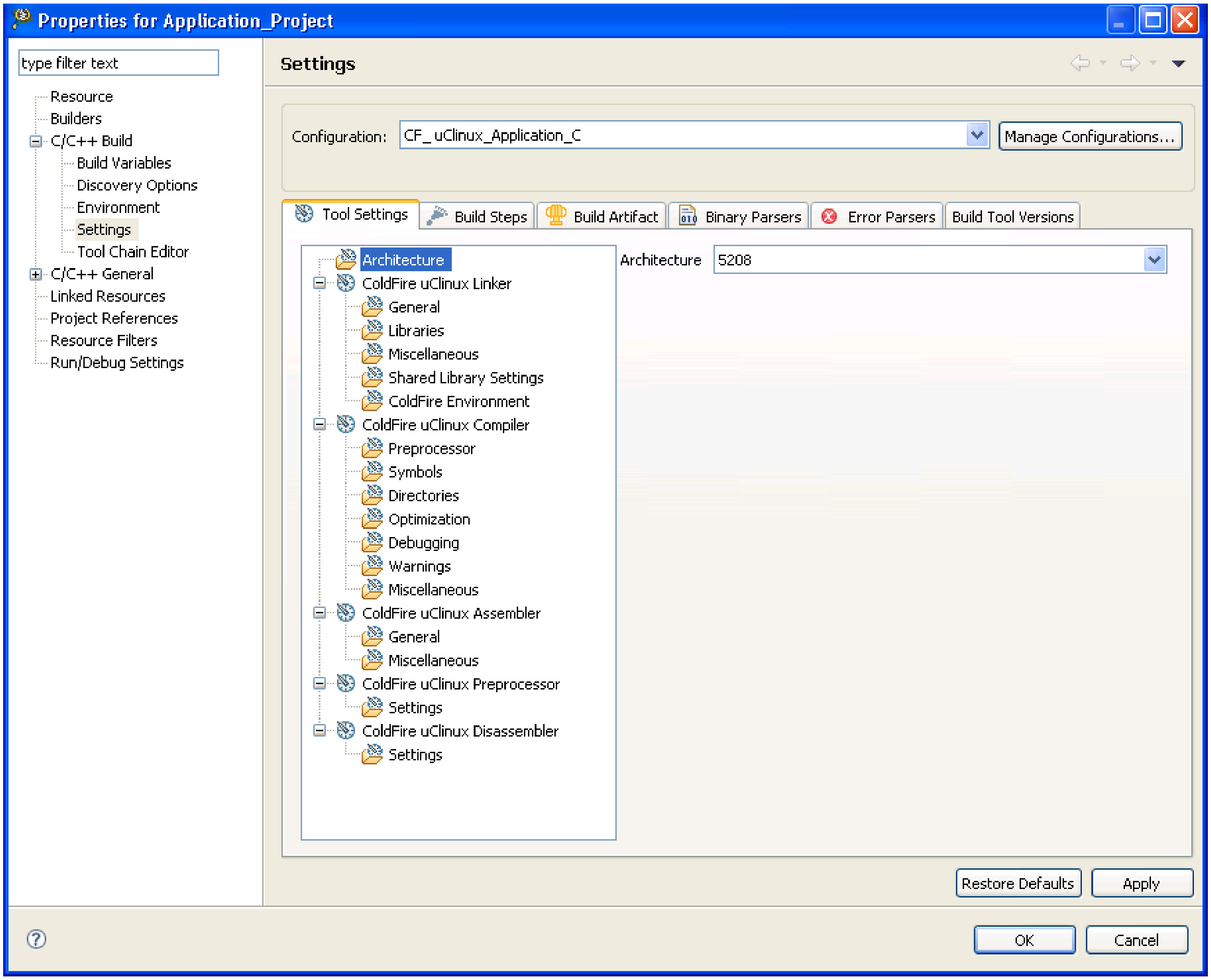
8.2 Restoring Build Properties

If you modify a build configuration that the CodeWarrior wizard generates, you can restore that configuration to its default state. You might want to restore the build properties in order to have a factory-default configuration, or to revert to a last-known working build configuration. To undo your modifications to build properties, click the **Restore Defaults** button at the bottom of the **Properties** window.

This changes the values of the options to the absolute default of the toolchain. By default, the toolchain options are blank.

8.3 Build Properties for Linux/uClinux Project

The **Properties for** <*project*> window shows the corresponding build properties for a uClinux project.



Configuration: CF_uClinux_Application_C

Architecture 5208

Figure 8-2. Build Properties - uClinux

The table below lists the build properties specific to developing software for uClinux.

The properties that you specify in these panels apply to the selected build tool on the **Tool Settings** page of the **Properties for** <*project*> window.

Table 8-1. Build Properties for uClinux Project

Build Tool	Build Properties Panels
Architecture	Architecture
ColdFire uClinux Linker	ColdFire uClinux Linker > General
	ColdFire uClinux Linker > Libraries
	ColdFire uClinux Linker > Miscellaneous
	ColdFire uClinux Linker > Shared Library Settings
	ColdFire uClinux Linker > ColdFire Environment
ColdFire uClinux Compiler	ColdFire uClinux Compiler > Preprocessor

Table continues on the next page...

Table 8-1. Build Properties for uClinux Project (continued)

Build Tool	Build Properties Panels
	ColdFire uClinux Compiler > Symbols
	ColdFire uClinux Compiler > Directories
	ColdFire uClinux Compiler > Optimization
	ColdFire uClinux Compiler > Debugging
	ColdFire uClinux Compiler > Warnings
	ColdFire uClinux Compiler > Miscellaneous
ColdFire uClinux Assembler	ColdFire uClinux Assembler > General
	ColdFire uClinux Assembler > Miscellaneous
ColdFire uClinux Preprocessor	ColdFire uClinux Preprocessor > Settings
ColdFire uClinux Disassembler	ColdFire uClinux Disassembler > Settings

8.3.1 Architecture

Use this panel to select the ColdFire uClinux architecture for the build.

The table below lists and describes the options in the **Architecture** panel.

Table 8-2. Tool Settings - Architecture Options

Option	Architecture
Architecture	Specify which architecture variant is used by the target.

8.3.2 ColdFire uClinux Linker

Use this panel to specify the command, options, and expert settings for the build tool linker. Additionally, the Linker tree control includes the general, libraries, and search path settings.

The table below lists and describes the options for the **ColdFire uClinux Linker** panel.

Table 8-3. tool Settings - ColdFire uClinux Linker Options

Option	Description
Command	Default: m `68k-uclinux-g++`
All options	Shows the actual command line the linker will be called with.

Table continues on the next page...

Table 8-3. tool Settings - ColdFire uClinux Linker Options (continued)

Option	Description
Expert settings	Default: "$ {CFGCCUCInstallDir}/${COMMAND}" $ {FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}$ {OUTPUT} ${INPUTS}
Command line pattern	

8.3.2.1 ColdFire uClinux Linker > General

Use this panel to specify general settings for the ColdFire uClinux linker.

The table below lists and describes the general options for the **ColdFire uClinux Linker** panel.

Table 8-4. Tool Settings - ColdFire uClinux Linker > General Options

Option	Description
Do not use standard start files (-nostartfiles)	Check if you do not want to use the standard system startup files when linking. The standard system libraries are used by default, unless `-nostdlib` or `-nodefaultlibs` is used.
Do not use default libraries (-nodefaulltlibs)	Check if you do not want to use the standard system libraries when linking. Only the libraries you specify will be passed to the linker. The standard startup files are used normally, unless `-nostartfiles` is used. The compiler may generate calls to memcmp, memset, memcpy and memmove. These entries are usually resolved by entries in `libc`. These entry points should be supplied through some other mechanism when this option is specified.
No startup or default libs (-nostdlib)	Check if you do not want to use the standard system startup files or libraries when linking. No startup files and only the libraries you specify will be passed to the linker. The compiler may generate calls to `memcmp`, `memset`, `memcpy`, and `memmove`. These entries are usually resolved by entries in `libc`. These entry points should be supplied through some other mechanism when this option is specified. One of the standard libraries bypassed by `-nostdlib` and `-nodefaultlibs` is `libgcc.a`, a library of internal subroutines that GCC uses to overcome shortcomings of particular machines, or special needs for some languages.
Omit all symbol information (-s)	Check if you do not want to remove all symbol table and relocation information from the executable.
No shared libraries (-static)	Check to prevent linking against or with the shared libraries for systems that support dynamic linking. On other systems, this option has no effect.

8.3.2.2 ColdFire uClinux Linker > Libraries

Use this panel to specify library settings for the **ColdFire uClinux Linker** . You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The table below lists and describes the libraries options for the ColdFire uClinux linker.

Table 8-5. ColdFire uClinux Linker > Libraries

Option	Description
Libraries (-l)	Specifies the linker command-line switches for any libraries that you want the IDE to include in the GCC command line for each file in the project.
Library search path (-L)	Specifies the search pathname of libraries or other resources related to the project.

8.3.2.3 ColdFire uClinux Linker > Miscellaneous

Use this panel to specify miscellaneous settings for the **ColdFire uClinux Linker** .

The table below lists and describes the miscellaneous options for **ColdFire uClinux Linker** .

Table 8-6. ColdFire uClinux Linker > Miscellaneous Options

Option	Description
Linker flags	Specify the other required linker flag options.
Other options (-Xlinker [option])	Pass an option to the linker. You can use this option to supply system-specific linker options which GCC does not know how to recognize. If you want to pass an option that takes a separate argument, you must use `-Xlinker` twice, once for the option and once for the argument. For example, to pass `-assert definitions`, you must write `-Xlinker -assert -Xlinker definitions`. It does not work if you write `-Xlinker "-assert definitions` because this passes the entire string as a single argument, which is not what the linker expects. When using the GNU linker, it is usually more convenient to pass arguments to linker options using the `option=value` syntax than as separate arguments.

Table continues on the next page...

Table 8-6. ColdFire uClinux Linker > Miscellaneous Options (continued)

Option	Description
	For example, you can specify `-Xlinker -Map=output.map` rather than `-Xlinker -Map -Xlinker output.map`. Other linkers may not support this syntax for command-line options.
Other objects	Specify the other required object options.

8.3.2.4 ColdFire uClinux Linker > Shared Library Settings

Use this panel to specify shared library settings for the **ColdFire uClinux Linker** .

The table below lists and describes the shared library settings for ColdFire uClinux linker.

Table 8-7. ColdFire uClinux Linker > Shared Library Settings Options

Option	Description
Shared (-shared)	Check to build shared versions of libraries, if shared libraries are supported on the target platform.
Shared object name (-Wl, -soname=)	Specifies the shared object name for the shared library.
Import Library name (-Wl, --out-implib=)	Specifies the import library name. The linker will create the file which will contain an import lib corresponding to the DLL the linker is generating. This import lib (which should be called *.dll.a or *.a may be used to link clients against the generated DLL; this behaviour makes it possible to skip a separate dlltool import library creation step. This option is specific to the i386 PE targeted port of the linker.
DEF file name (_Wl, --output-def=)	Species the name of the `.def` file to be created by dlltool.

8.3.2.5 ColdFire uClinux Linker > ColdFire Environment

Use this panel to specify environment settings for the **ColdFire uClinux Linker** .

The table below lists and describes the shared ColdFire environment settings for ColdFire uClinux linker.

Table 8-8. ColdFire uClinux Linker > ColdFire Environment Options

Option	Description
Map File	Specify the file that will be written as a linker memory map file.
LCF File	Read the specified linker script file.

8.3.3 ColdFire uClinux Compiler

Use this panel to specify the command, options, and expert settings for the build tool compiler. Additionally, the Compiler tree control includes the general, libraries, and search path settings.

The table below lists and describes the options for ColdFire uClinux compiler.

Table 8-9. Tool Settings - ColdFire uClinux Compiler Options

Option	Description
Command	Default: `m68k-uclinux-gcc`
All options	Shows the actual command line the compiler will be called with.
Expert settings Command line pattern	Default: " `${CFGCCUCInstallDir}/${COMMAND}" ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`

8.3.3.1 ColdFire uClinux Compiler > Preprocessor

Use this panel to specify the preprocessor behavior. You can specify the file paths and define macros for the preprocessor.

The table below lists and describes the options for the **ColdFire uClinux Compiler** panel.

Table 8-10. ColdFire uClinux Compiler > Preprocessor Options

Option	Description
Do not search system directories (-nostdinc)	Select if you do not want to search the standard system directories for header files. Only the directories you have specified with `-I' options and the directory of the current file, if appropriate are searched.
Preprocess only (-E)	Select if you do not want to compile, assemble, or link.

8.3.3.2 ColdFire uClinux Compiler > Symbols

Use this panel to specify code- and symbol-generation options for the **ColdFire uClinux Compiler** .

The table below lists and describes the options for the **ColdFire uClinux Compiler** panel.

Table 8-11. ColdFire uClinux Compiler > Symbols Options

Option	Description
Defined symbols (-D)	Display only defined symbols for each object file.
Undefined symbols (-U)	Display only undefined symbols; those external to each object file.

8.3.3.3 ColdFire uClinux Compiler > Directories

Use this panel to specify include path directories for the **ColdFire uClinux Compiler** .

The table below lists and describes the options for the **Directories** panel.

Table 8-12. ColdFire uClinux Compiler > Directories Options

Option	Description
Include paths	Specify locations to header files. Append a list of directories to the standard directory list.

8.3.3.4 ColdFire uClinux Compiler > Optimization

Use this panel to control compiler optimizations. The compiler's optimizer can apply any of its optimizations in either global or non-global optimization mode. You can apply global optimization at the end of the development cycle, after compiling and optimizing all source files individually or in groups.

The table below lists and describes the options for the **Optimization** panel.

Table 8-13. ColdFire uClinux Compiler > Optimization Options

Option	Description
Optimization Level	Specify the optimizations that you want the compiler to apply to the generated object code: • **None (-O0) -** Disable optimizations. This setting is equivalent to specifying the `-O0` command-line option. The compiler generates unoptimized, linear assembly-language code. • **Optimize (-O1) -** The compiler performs all target-independent (that is, non-parallelized) optimizations, such as function inlining. This setting is equivalent to specifying the `-O1` command-line option. The compiler omits all target-specific optimizations and generates linear assembly-language code. • **Optimize more (-O2) -** The compiler performs all optimizations (both target-independent and target-specific). This setting is equivalent to specifying the `-O2` command-line option. The compiler outputs optimized, non-linear, parallelized assembly-language code. • **Optimize most (-O3) -** The compiler performs all the level **2** optimizations, then the low-level optimizer performs global-algorithm register allocation. This setting is equivalent to specifying the `-O3` command-line option. At this optimization level, the compiler generates code that is usually faster than the code generated from level **-O2** optimizations.
Other optimization flags	Specify other optimization flags, like -Os (optimize for size) or specific optimizations if you would like to do a "fine-tuning" of optimizations.

8.3.3.5 ColdFire uClinux Compiler > Debugging

Use this panel to specify whether to generate symbolic information for debugging the build target.

The table below lists and describes the options for the **Debugging** panel.

Table 8-14. ColdFire uClinux Compiler > Debugging Options

Option	Description
Debug Level	Generates a compiled file containing debugging information. By default, it will be generated in DWARF-2 format.
Other debugging flags	Allows specifying other debugging options, like changing the format of debugging information or the level.
Generate gprof information (-pg)	Generate extra code to write profile information suitable for the analysis program `gprof`. You must use this option when compiling the source files you want data about, and you must also use it when linking.
Generate prof information (-p)	Generate extra code to write profile information suitable for the analysis program `prof`. You must use this option when compiling the source files you want data about, and you must also use it when linking.

8.3.3.6 ColdFire uClinux Compiler > Warnings

Use this panel to control how the ColdFire compiler formats the listing file, as well as the error and warning messages.

The table below lists and describes the options for the **Warnings** panel.

Table 8-15. ColdFire uClinux Compiler > Warnings

Option	Description
Check syntax only (-fsyntax-only)	Check the option if you want to scan the code only for syntax errors and nothing else. Warnings are diagnostic messages that report constructions which are not inherently erroneous but which are risky or suggest there may have been an error. The following language-independent options do not enable specific warnings but control the kinds of diagnostics produced by GCC.
Pednatic (-pedantic)	Check is you want to issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any `-std' option used.
Pedantic warnings as errors (-pedantic-errors)	Check if you want warnings like `-pedantic`, except that errors are produced rather than warnings.
Inhibit all warnings (-w)	Check if you want to inhibit all warning messages.

Table continues on the next page...

Table 8-15. ColdFire uClinux Compiler > Warnings (continued)

Option	Description
All warnings (-Wall)	Check to enable all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros.
Warnings as errors (-Werror)	Check to make all warnings into hard errors. Source code which triggers warnings will be rejected.

8.3.3.7 ColdFire uClinux Compiler > Miscellaneous

Use this panel to specify miscellaneous settings for the **ColdFire uClinux Compiler**.

The table below lists and describes the miscellaneous options for **ColdFire uClinux Linker**.

Table 8-16. ColdFire uClinux Compiler > Miscellaneous

Option	Description
Other flags	Specify additional command line options for the compiler; type in custom flags that are not otherwise available in the UI.
Verbose (-v)	Check to enable the verbose mode. Print out GNU CPP's version number at the beginning of execution, and report the final form of the include path.
Support ANSI program (-ansi)	Check to specify the standard to which the code should conform.
No common symbols (-fno-common)	Check to control the placement of uninitialized global variables.

8.3.4 ColdFire uClinux Assembler

Use this panel to specify the command, options, and expert settings for the build tool assembler. Additionally, the Assembler tree control includes the general and include file search path settings.

The table below lists and defines each option of the **ColdFire uClinux Assembler** panel.

Table 8-17. ColdFire uClinux Assembler

Option	Description
Command	Enter the command line arguments for the GCC assembler in the **Command Line Arguments** text box. The contents of this text box are passed as command-line switches in the gcc command line for each file in your project as they are assembled. Shows the location of the assembler executable file. Default value is: `m68k-uclinux-as`
All options	Shows the actual command line the assembler will be called with.
Expert Settings	
Command line pattern	Default value is: `"${CFGCCUCInstallDir}/${COMMAND}" ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}`

8.3.4.1 ColdFire uClinux Assembler > General

Use this panel to specify additional files the **ColdFire Assemble** r should use. You can specify multiple additional libraries and library search paths. Also, you can change the order in which the IDE uses or searches the libraries.

The table below lists and defines each option of the **General** panel.

Table 8-18. ColdFire uClinux Assembler > General

Option	Description
Assembler flags	Specify options that will be passed to the assembler.
Include paths (-I)	Use this option to add a path to the list of directories as searches for files specified in `.include` directives.
Suppress warnings (-W)	Check to inhibit all warning messages.
Announce version (-v)	Check to find out what version of assembler is running.

8.3.4.2 ColdFire uClinux Assembler > Miscellaneous

Use this panel to specify miscellaneous settings for the **ColdFire uClinux Assembler** .

The table below lists and defines each option of the **Miscellaneous** panel.

Table 8-19. ColdFire uClinux Assembler > Miscellaneous

Option	Description
Other options (-Xassembler)	Provide an option to the assembler. You can use this to supply system-specific assembler options which GCC does not know how to recognize. If you want to pass an option that takes an argument, you must use `-Xassembler` twice, once for the option and once for the argument.

8.3.5 ColdFire uClinux Preprocessor

Use this panel to specify preprocessor behavior and define macros.

The table below lists and defines each option of the **Preprocessor** panel.

Table 8-20. ColdFire uClinux Preprocessor

Option	Description
Command	Provide the command line arguments for the GCC assembler in the **Command Line Arguments** text box. The contents of this text box are passed as command-line switches in the gcc command line for each file in your project as they are assembled. Shows the location of the linker executable file. Default value is: `m68k-uclinux-gcc`
All options	Shows the actual command line the preprocessor will be called with.
Expert Settings	Default value is: `"${CFGCCUCInstallDir}/${COMMAND}" ${FLAGS} ${INPUTS}`
Command line pattern	

8.3.5.1 ColdFire uClinux Preprocessor > Settings

Use this panel to specify preprocessor behavior.

The table below lists and defines each option of the **Settings** panel.

Table 8-21. ColdFire uClinux Preprocessor > Settings

Option	Description
Handle Directives Only	Select to handle directives only.
Print Header File Names	Select if you want to print header filenames.

8.3.6 ColdFire uClinux Disassembler

Use this panel to specify the command, options, and expert settings for the **ColdFire uClinux Disassembler**.

The table below lists and defines each option of the **Settings** panel.

Table 8-22. ColdFire uClinux Disassembler

Option	Description
Command	You can enter the command line arguments for the GCC disassembler in the **Command** text box. The contents of this text box are passed as command-line switches in the gcc command line for each file in your project as they are assembled. Shows the location of the linker executable file. Default value is: `m68k-uclinux-objdump`.
All options	Shows the actual command line the preprocessor will be called with.
Expert Settings	Default value is: `"${CFGCCUCInstallDir}/${COMMAND}" ${FLAGS} ${INPUTS}`
Command line pattern	

8.3.6.1 ColdFire uClinux Disassembler > Settings

Use this panel to control how the disassembler formats the listing file, as well as error and warning messages. You can specify verbosity of messages, whether to show headers, core modules, extended mnemonics, addresses, object or source code, data modules, exception tables, and debug information.

The table below lists and defines each option of the **Settings** panel.

Table 8-23. ColdFire uClinux Disassembler

Option	Description
Disassemble All Section Content	Disassembles all section content and sends the output to a file. This command is global and case-sensitive.
Disassemble Executable Section Content	Disassembles all executable content and send output to a file.
Intermix Source Code with Disassembly	Turn jbsr into jsr.
Display All Header Content	Display the contents of all headers.
Display Archive Header Information	Display archive header information.
Display Overall File Header Content	Display the contents of the overall file header.
Display Full Section Content	Display the full section of the file.
Display Debug Information	Display debug information in the object file.
Display Debug Information Using ctag Style	Display debug information using the ctags style.
Display STABS Information	Displays any STABS information in the file, in raw form.
Display DWARF Information	Displays any DWARF information in the file.
Display Symbol Table Content	Displays the contents of the symbol tables.
Display Dynamic Symbol Table Content	Displays the contents of the dynamic symbol table.
Display Relocation Entries	Displays the relocation entries in the file.
Display Dynamic Relocation Entries	Displays the dynamic relocation entries in the file.

Chapter 9
Connections - HCS08 Architecture

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the HCS08 simulator or the target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the **Launch Configuration** panel. The **Launch Configuration** panel can be accessed by clicking on the **Edit** button located within the **Main** tab of the **Debug Configurations** dialog box. These options are:

- The **Connection Type** option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with simulation platforms and hardware devices that are part of the HCS08 device family.

The topics in this chapter are:

- P&E Full Chip Simulation
- P&E Hardware Interface Connection for HCS08

9.1 P&E Full Chip Simulation

This topic details the settings of the connections that interface the CodeWarrior debugger with the HCS08 simulator.

9.1.1 Create New Connection for Full Chip Simulation

A Full Chip Simulation (FCS) connection runs a complete simulation of all processor peripherals and I/O on your personal computer. Thus, when debugging an FCS project for a selected derivative it is not necessary to connect your PC with a Microcontrollers development or target board.

To change connection in the IDE, perform these steps.

1. Right-click on your Project > Properties.

   The Properties windows appears.

2. Select Run/Debug Settings and click on the New... button.
3. Select CodeWarrior Download and click OK.

   The Edit Configuration window appears.

4. By default the project and application is already set. Change the name of your connection if you wish. You will need to create a new Connection. Within the Connection section, click on the New... button.

   The New Connection Wizard will appear.

5. Open the CodeWarrior Bareboard Debugging and select Hardware or Simulator Connection. Click Next.
6. Give your new connection a name. For connection type, change the setting to P&E HCS08 FCS for Full Chip Simulation.
7. You will need to select the Target device. You can either select a pre-existing target or create a New target. When complete, click on the Finish button.
8. The wizard creates a simulator project for the HCS08 architecture according to your specifications. You can access and edit the project connections by right-click on your project > Debug As > Debug Configurations.

9.1.2 Module Options

The PEMicro menu includes the Full Chip Simulation options for the modules that have specialty commands associated with them for a chosen device.

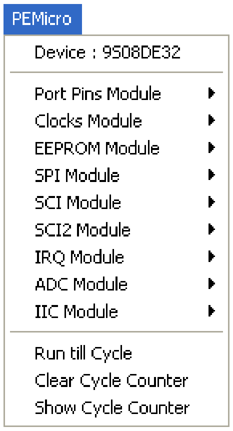


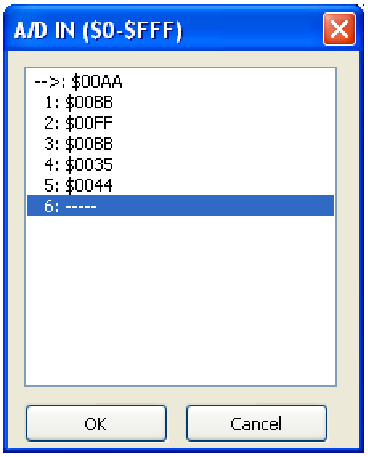
Figure 9-1. PEMicro Menu

The options available are:

- Analog-to-Digital Converter Module
- 16-Bit Analog-to-Digital Converter Module
- Clock Generation Module
- Digital-to-Analog Converter Module
- EEPROM Module
- Fault Detection and Shutdown Module
- Flash Module
- Flextimer Module
- High-Speed Analog Comparator Module Flextimer Module
- Inter-Integrated Circuit Module
- Interrupt Priority Controller Module
- External Interrupt (IRQ) Module
- Keyboard Interrupt Module
- Liquid Crystal Display Driver Module
- Modulo Timer Interrupt Module
- MSCAN Controller Module
- Programmable Delay Block Module
- Programmable Gain Amplifier Module
- Programmable Reference Analog Comparator Module
- Input/Output (I/O) Ports Module
- Serial Communications Interface Module
- Slave LIN Interface Controller (SLIC) Module
- Serial Peripheral Interface Module
- Timer Interface Module
- Time Of Day Module Option
- Universal Serial Bus (USB) Module
- Voltage Reference Module

9.1.2.1 Analog-to-Digital Converter Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Analog-to-Digital Conversion (ADC) module, including data input on all ADC channels, flag polling, interrupt operation, and the bus and CGMXCLK reference clock sources. FCS mode uses the buffered input structure to simulate the ADC inputs. You can queue up to 256 data values. To queue the ADC Input Data, use the `ADDI` command in the command prompt. If the data parameter is given, the value is placed into the next slot in the input buffer. Otherwise, if no parameter is provided, a window is displayed with the input buffer values. Input values can be entered while the window is open. An arrow points to the next value to be used as input to the ADC. The conversion takes place after a proper value is written to the ADC Status and Control register. Once the conversion occurs, the arrow moves to the next value in the ADC Buffer.

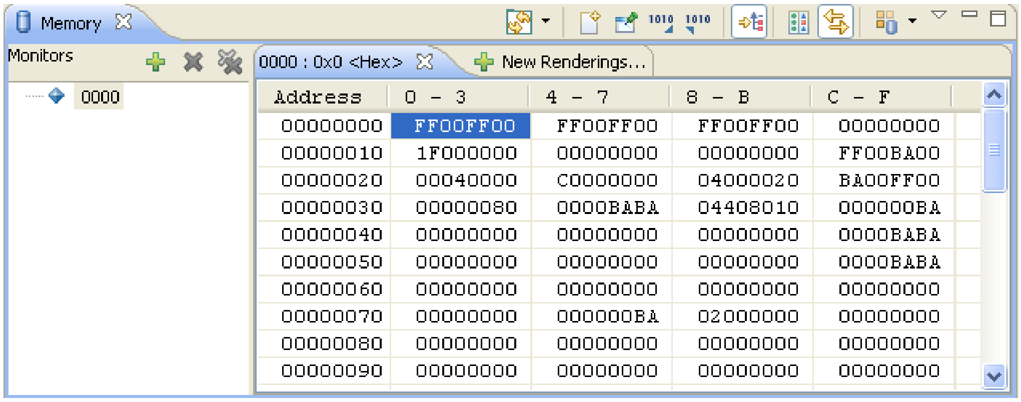

A/D IN ($0-$FFF)

```
-->: $00AA
  1: $00BB
  2: $00FF
  3: $00BB
  4: $0035
  5: $0044
  6: -----
```

Figure 9-2. ADC IN Buffer Display

The `ADCLR` command can be used at any point to flush the input buffer for the ADC simulation.

After the conversion is complete, the first queued value is passed from the data buffer into the ADC data register. It can be observed in the Memory window by displaying the memory location corresponding to the ADC data register.



Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 9-3. Memory Component Window

When the conversion is complete, FCS sets the appropriate flag. If interrupts are enabled, the Program Counter changes flow to the interrupt routine (as defined in the vector space of the MCU).

NOTE

For more information on ADC configuration, refer to the Freescale user manual for your microprocessor.

9.1.2.1.1 ADC Module Commands

The following commands are available for the HC08/HCS08 ADC Module.

9.1.2.1.2 ADDI Command

The `ADDI` command lets you input the data into the ADC converter. If a data parameter is given, the value is placed into the next slot in the input buffer. Otherwise, if no parameter is given, a window is displayed with the input buffer values. Input values can be entered while the window is open. An arrow points to the next value to be used by the ADC. The maximum number of input values is 256 bytes. Syntax

```
>gdi
ADDI [<n>]
```

Where: `<n>` The value to be entered into the next location in the input buffer. Example

```
>gdi
ADDI $55
```

Set the next input value to the ADDI to $55

```
>gdi ADDI
```

Pull up the data window with all the input values.

9.1.2.1.3 ADCLR Command

Use the `ADCLR` command to flush the input buffer for ADC simulation. This resets the input data buffer and clears out all values. Notice that if the ADC is currently using a value, this command does not prevent the ADC from using it.

NOTE

> Refer the `ADDI` command for information on how to access the input buffer of the ADC interface.

Syntax

```
>gdi ADCLR
```

Example

```
>gdi ADCLR
```

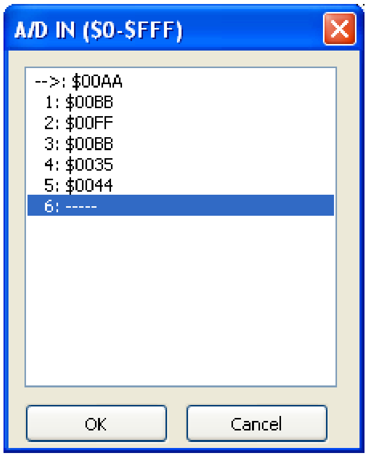
Clear the input buffer for ADC simulation.

9.1.2.2 16-Bit Analog-to-Digital Converter Module

The following commands are available for the HCS08 ADC16V1 Module.

9.1.2.2.1 ADDI Command

The ADDI command allows the user to input the data into the ADC16 converter. If a data parameter is given, the value is placed into the next slot in the input buffer. If no data parameter is given, a window is displayed with the input buffer values shown in the followimng figure. Input values can be entered while the window is open. An arrow points to the next value that will be used by the ADC16. The maximum number of input values is 256 bytes.



```
-->: $00AA
  1: $00BB
  2: $00FF
  3: $00BB
  4: $0035
  5: $0044
  6: -----
```

Figure 9-4. ADC16 IN Buffer Display

After the conversion is complete, the first queued value is passed from the data buffer into the ADC16 data register. It can be observed in the memory window by displaying the memory location corresponding to the ADC16 data register. Syntax

```
>gdi ADDI <n>
```

Where: `<n>` The value to be entered into the next location in the input buffer. Example

```
>gdi ADDI $55
```

Set the next ADDI input value to $55

```
>gdi ADDI
```

Pull up the data window with all the input values.

9.1.2.2.2 ADCLR Command

Use the ADCLR command to flush the input buffer for ADC16 simulation. This resets the input data buffer and clears out all values. Note that if the ADC16 is currently using a value, this command does not prevent the ADC16 from using it. Refer the ADDI command for information on how to access the input buffer of the ADC16 interface.

Syntax

```
>gdi ADCLR
```

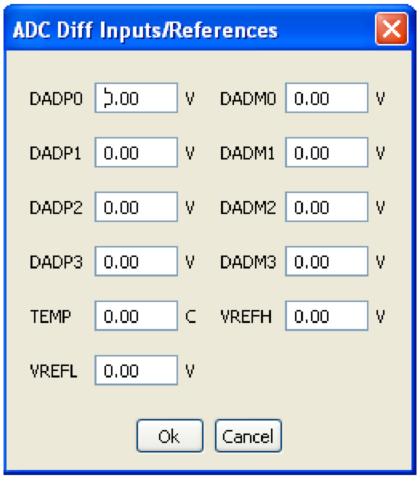
Example

```
>gdi ADCLR
```

Clear the input buffer for ADC simulation.

9.1.2.2.3 ADDID Command

The ADDID command allows the user to input the differential data into the ADC16 converter. A window is displayed that allows the user to input differential values. The user may specify the differential data of the Differential Analog Channel (DADx) input voltages, the Temperature Sensor (TEMP) input, and the Voltage References (VREF).



ADC Diff Inputs/References

DADP0	0.00	V	DADM0	0.00	V
DADP1	0.00	V	DADM1	0.00	V
DADP2	0.00	V	DADM2	0.00	V
DADP3	0.00	V	DADM3	0.00	V
TEMP	0.00	C	VREFH	0.00	V
VREFL	0.00	V			

Figure 9-5. ADC Differential Data Display

When using FCS, the ADDID command shows the simulated differential inputs to the ADC16 module. Syntax

```
>gdi ADDID
```

Example

```
>gdi ADDID
```

Pull up the differential data window

9.1.2.3 Clock Generation Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Clock Generation Module (ICG), including:

- Phase Locked Loop (PLL) generation
- Automatic lock detection
- Interrupt
- Acquisition
- Tracking
- Flag polling

FCS mode uses a simulated External Oscillator Frequency change command (`XTAL`) lets you input the desired XTAL value. To check the current value of the External Oscillator, Bus Frequency and CGMXCLK Frequency, open the HCS08FCS menu and select Clocks Module > Show MCU Clocks.

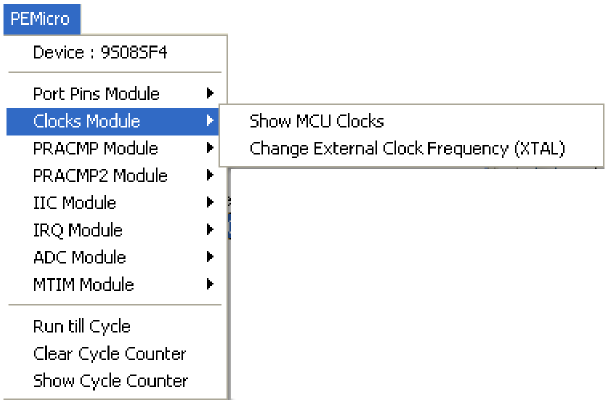
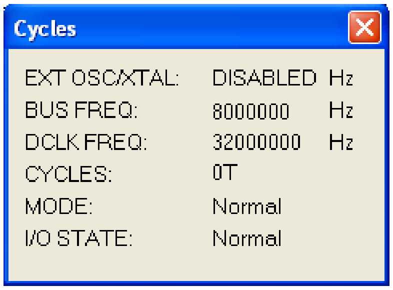


Figure 9-6. Clocks Module Extended Menu

Once you select the MCU Clocks menu, the Cycles window displays all of the aforementioned Clock Frequencies, or you can select the Show Cycle Counter option within the FCS menu to get the same window.



```
EXT OSC/XTAL:    DISABLED  Hz
BUS FREQ:        8000000   Hz
DCLK FREQ:       32000000  Hz
CYCLES:          0T
MODE:            Normal
I/O STATE:       Normal
```

Figure 9-7. Frequency Display

Within the FCS menu, you can select the Run till Cycle option, which lets you begin code execution and stop execution when the specified cycle count is reached. Note that the parameter given is not the number of cycles that executed, but rather the total cycle-count of the simulator (displayed in the Register Window).

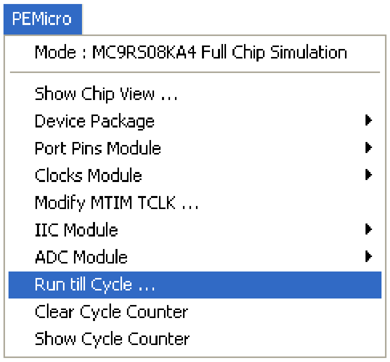
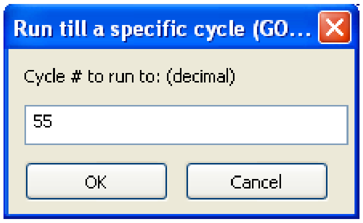


Figure 9-8. Run till Cycle Command

This command is extremely useful for verifying specific timings of a given event, running until a given event is complete, or just before it completes to enable stepping through the event or any application where cycle-timed execution is desired.

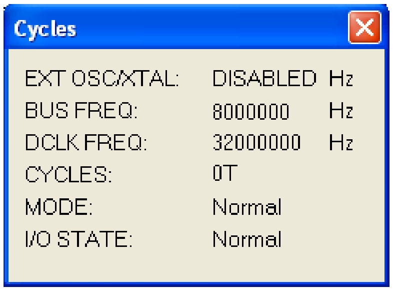

Run till a specific cycle (GO...

Cycle # to run to: (decimal)

55

Figure 9-9. Run till a specific cycle Dialog Box

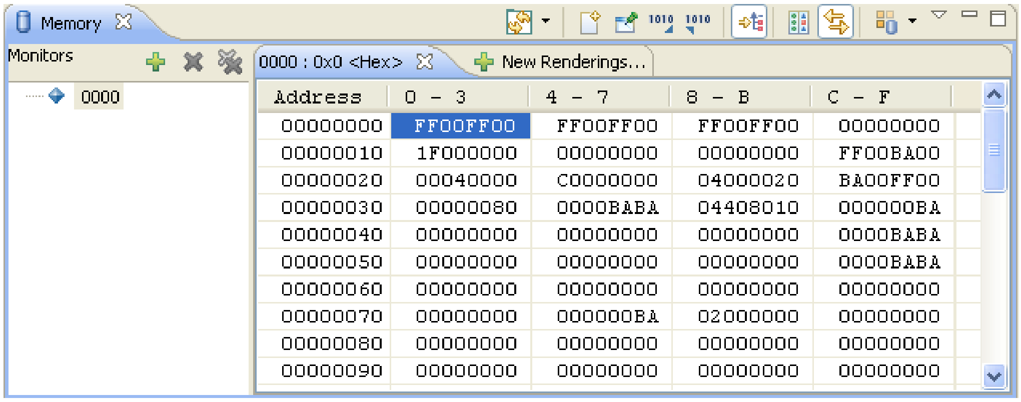
You can also select the Clear Cycle Counter option within the FCS menu, which clears the cycle counter. If you select the Show Cycle Counter option within the FCS menu, you can check to make sure that the cycle counter is zero.


Cycles

EXT OSC/XTAL:	DISABLED	Hz
BUS FREQ:	8000000	Hz
DCLK FREQ:	32000000	Hz
CYCLES:	0T	
MODE:	Normal	
I/O STATE:	Normal	

Figure 9-10. Cycles Dialog Box with Cleared Counter

Once the ICG is properly configured, you can monitor the status of the PLL by polling the corresponding flag. If PLL interrupt is enabled, FCS jumps to an appropriate subroutine, as long as the interrupt vector is properly defined. To observe the flag going up as a result of the corresponding CPU event, situate your Memory window on the memory location of the ICG Status and Control register.



```
Address    0 - 3      4 - 7      8 - B      C - F
00000000   FF00FF00   FF00FF00   FF00FF00   00000000
00000010   1F000000   00000000   00000000   FF00BA00
00000020   00040000   C0000000   04000020   BA00FF00
00000030   00000080   0000BABA   04408010   000000BA
00000040   00000000   00000000   00000000   0000BABA
00000050   00000000   00000000   00000000   0000BABA
00000060   00000000   00000000   00000000   00000000
00000070   00000000   000000BA   02000000   00000000
00000080   00000000   00000000   00000000   00000000
00000090   00000000   00000000   00000000   00000000
```

Figure 9-11. Memory Window

For more information on how to properly configure Clock Generation, refer to the Freescale reference manual for your microprocessor.

9.1.2.3.1 Clock Generation Module Commands

The following commands are available for the HC08/HCS08 Clock Generation Module.

9.1.2.3.1.1 XTAL Command

Use the `XTAL` command to change the value of the simulated external oscillator. This in turn affects the input to the PLL/DCO, and therefore the bus frequency. The P&E simulator is a cycle-based simulator, so changing the XTAL value does not affect the speed of simulation. It does, however, affect the ratio in which peripherals receive cycles. Certain peripherals that run directly from the XTAL will run at different speeds than those that run from the bus clock.

9.1.2.3.1.2 Syntax

```
>gdi XTAL <n>
```

where, `<n>`, by default, is a hexadecimal number, representing the simulated frequency of an external oscillator. Adding the suffix `` `t' `` to the `'n'` parameter forces the input value to be interpreted as base 10.

Example

```
>gdi
XTAL
```

Brings up an input window. The default base for this input value is 10. However, this value can be forced to a hexadecimal format through use of the suffix `` `h' ``.

9.1.2.4 Digital-to-Analog Converter Module

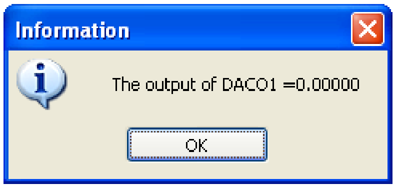
In Full Chip Simulation (FCS) Mode, this module lets you simulate all the functionality of the 5-bit Digital-to-Analog Converter (DAC) module. This module provides 32 distinct selectable voltage levels through the use of a 32-tap resistor ladder network and a 32-to-1 multiplexer. Each DAC module output can be routed to an HSCMP input.

9.1.2.4.1 Digital-to-Analog User Commands

The following DAC commands are available for the HCS08.

9.1.2.4.2 SHOWDACO1 Command

The SHOWDACO1 command displays the DAC Output dialog box shown in the following figure.



The output of DACO1 =0.00000

Figure 9-12. Figure DAC Output Dialog Box

Syntax

```
>gdi SHOWDACO1
```

Example

```
>gdi SHOWDACO1
```

Show DAC Output Dialog Box

9.1.2.5 EEPROM Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of EEPROM module including sector erase abort, burst programming capability, security feature, flexible block protection and vector redirection, and command interface for fast program and erase operation.

9.1.2.5.1 EEPROM User Commands

The following `EEPROM` commands are available for the HCS08.

9.1.2.5.2 EEPROM<x> Command

The `EEPROM<x>` command simulates changing of the EEPROM page for devices that have paged EEPROM.

Syntax

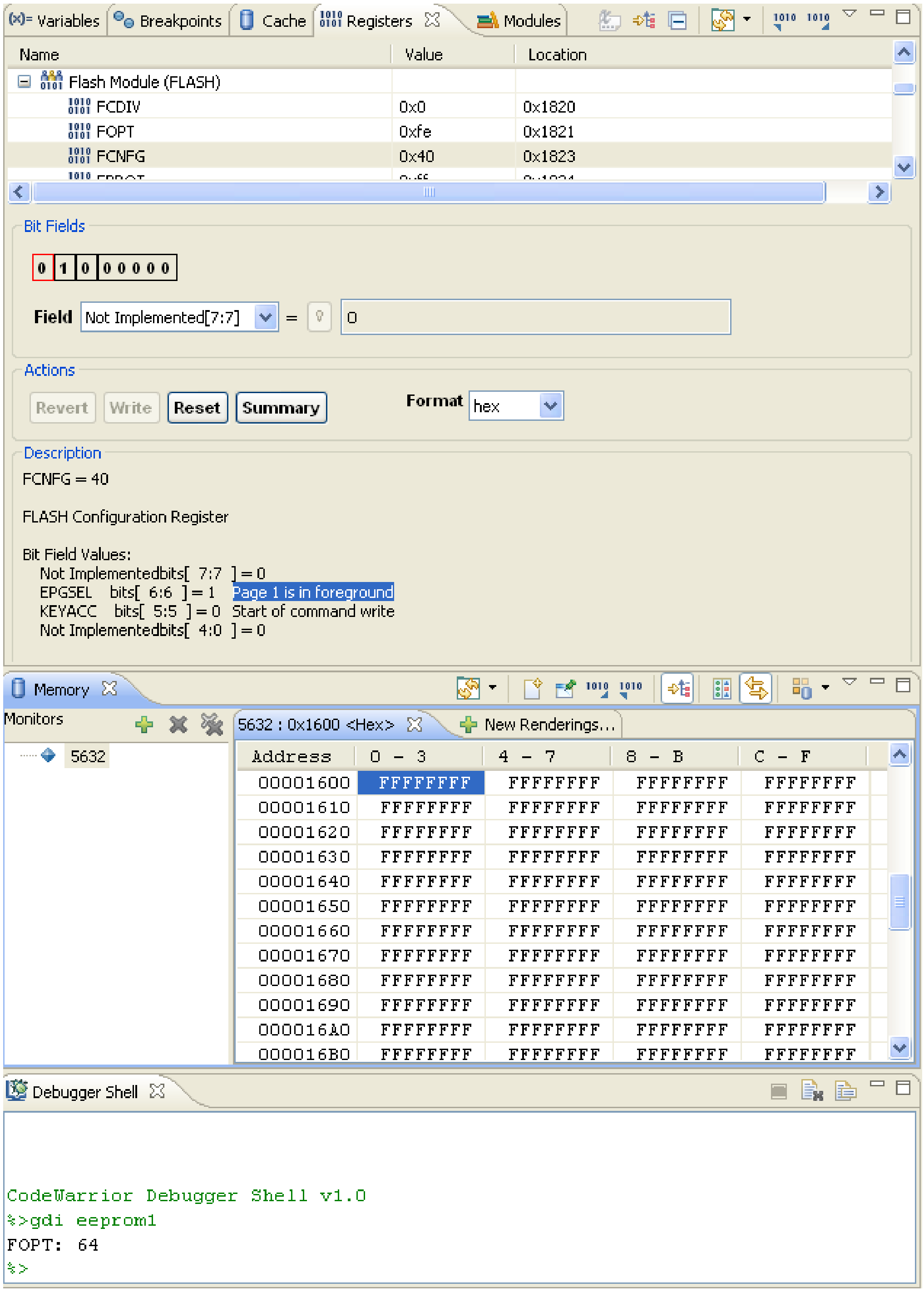
```
>gdi EEPROM<x>
```

Where: `<x>` is the letter representing corresponding EEPROM page number

Example

```
> gdi EEPROM1
```

Simulate change to EEPROM page 1.



Name	Value	Location
Flash Module (FLASH)		
FCDIV	0x0	0x1820
FOPT	0xfe	0x1821
FCNFG	0x40	0x1823

Bit Fields

0 1 0 0 0 0 0 0

Field Not Implemented[7:7] = 0

Actions

Format hex

Description

FCNFG = 40

FLASH Configuration Register

```
Bit Field Values:
   Not Implementedbits[  7:7  ] = 0
   EPGSEL    bits[  6:6  ] = 1  Page 1 is in foreground
   KEYACC    bits[  5:5  ] = 0  Start of command write
   Not Implementedbits[  4:0  ] = 0
```

```
Address   0 - 3     4 - 7     8 - B     C - F
00001600  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001610  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001620  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001630  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001640  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001650  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001660  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001670  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001680  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001690  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
000016A0  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
000016B0  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
```

```
CodeWarrior Debugger Shell v1.0
%>gdi eeprom1
FOPT:  64
%>
```

Figure 9-13. Example of using EEPROM<x>Command

9.1.2.6 Fault Detection and Shutdown Module

In Full Chip Simulation (FCS) Mode, this module lets you simulate all the functionality of the Fault Detection and Shutdown (FDS) module. When a fault condition occurs, the module provides a mechanism to immediately place port pins into a pre-defined state; the output pin of FDS can be configured as output 0, output 1, high impedance, or bypass during shutdown. The module can configure up to 8 fault input sources and control up to 8 port pins.

9.1.2.7 Flash Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of Flash module including sector erase abort, burst programming capability, security feature, flexible block protection and vector redirection, and command interface for fast program and erase operation.

9.1.2.7.1 Flash User Commands

The following Flash commands are available for the HCS08.

9.1.2.7.2 PPAGE <x> Command

The `PPAGE <x>` command simulates changing of Flash PPAGE for devices that have paged FLASH.

Syntax

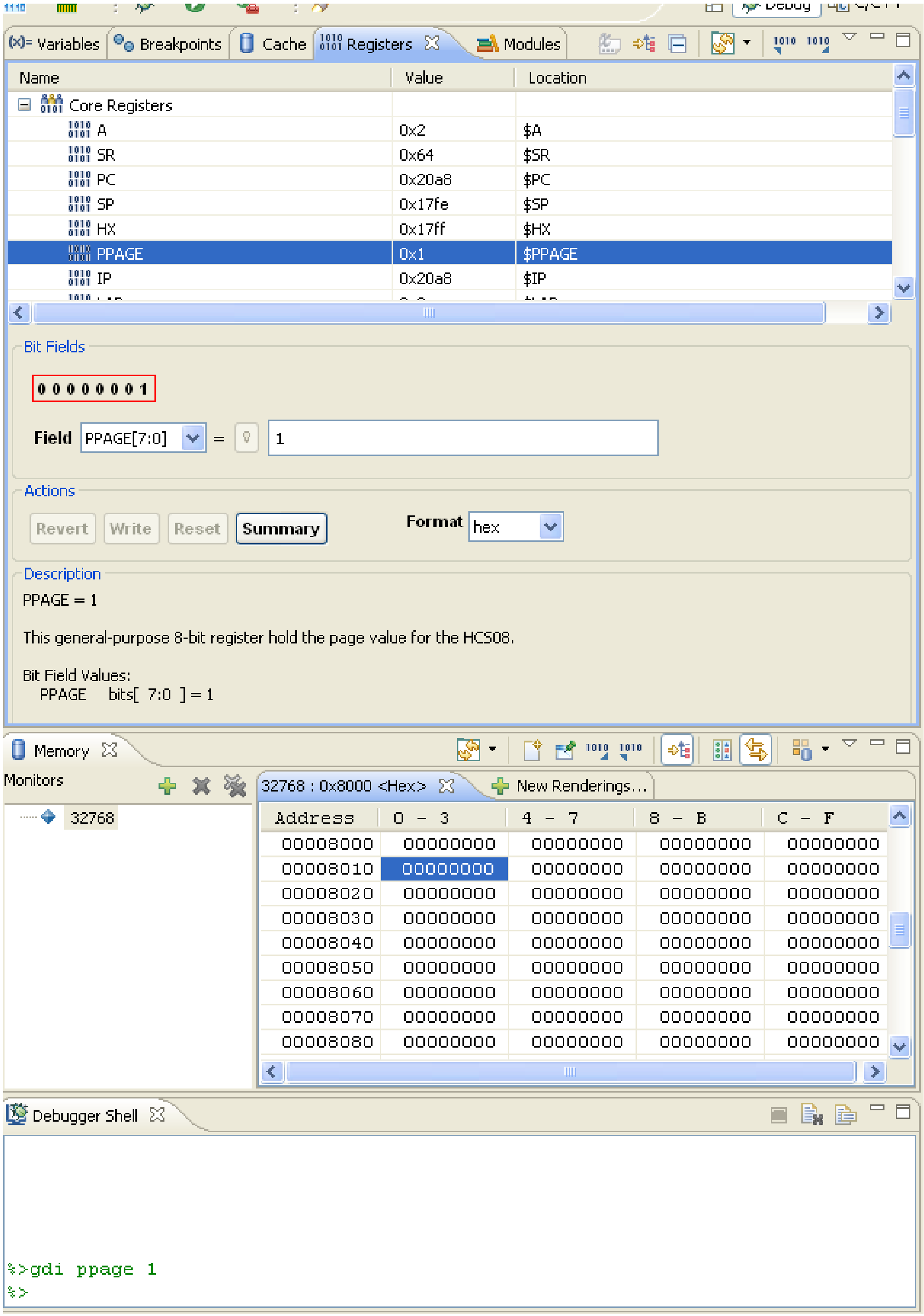
```
>gdi PPAGE <x>
```

Where: `<x>` is the letter representing corresponding PPAGE number

Example

```
>gdi PPAGE 1
```

Simulate change to PPAGE 1.



Name	Value	Location
Core Registers		
A	0x2	$A
SR	0x64	$SR
PC	0x20a8	$PC
SP	0x17fe	$SP
HX	0x17ff	$HX
PPAGE	0x1	$PPAGE
IP	0x20a8	$IP

Bit Fields

0 0 0 0 0 0 0 1

Field PPAGE[7:0] = 1

Actions

Format hex

Description

PPAGE = 1

This general-purpose 8-bit register hold the page value for the HCS08.

Bit Field Values:
  PPAGE   bits[7:0] = 1

32768 : 0x8000 <Hex>

```
32768

Address    0 - 3     4 - 7     8 - B     C - F
00008000  00000000  00000000  00000000  00000000
00008010  00000000  00000000  00000000  00000000
00008020  00000000  00000000  00000000  00000000
00008030  00000000  00000000  00000000  00000000
00008040  00000000  00000000  00000000  00000000
00008050  00000000  00000000  00000000  00000000
00008060  00000000  00000000  00000000  00000000
00008070  00000000  00000000  00000000  00000000
00008080  00000000  00000000  00000000  00000000
```

```
%>gdi ppage 1
%>
```

Figure 9-14. Example of using PPAGE <x> command

9.1.2.8 Flextimer Module

In Full Chip Simulation (FCS) Mode, this option lets you simulate all the functionality of the Flextimer (FTMV2) module, including:

- Input capture/output compare
- Pulse width modulation
- Internal or external clock input
- Free running or modulo up count operation
- Flag polling
- Interrupt enabled mode of operation.
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal (with independent control of both edges of PWM signal)
- The FTM channels can operate as pairs with equal outputs, pairs with complimentary outputs, or independent channels (with independent outputs)
- The dead-time insertion is available for each complementary pair
- Generation of triggers (hardware trigger)

9.1.2.9 High-Speed Analog Comparator Module

In Full Chip Simulation (FCS) mode, this option lets you simulate all the functionality of the High-Speed Analog Comparator (HSCMP) module, including data input on all HSCMP channels, flag polling, and interrupt operation, as well as output connection to PDB input triggers. The user can utilize either the HSCMP Inputs display form or command-line commands to provide inputs to the HSCMP module.

9.1.2.9.1 High-Speed Analog Comparator User Commands

The following commands are available for the HCS08 HSCMP Module.

9.1.2.9.2 HSC<x>INPUT<y> Command

The HSC<x>INPUT<y> command lets you input a voltage value for the external analog input CIN<y>. For HCS08 devices that have more than one HSCMP module, <x> is the number representing the corresponding module. For HCS08 devices that only have one HSCMP module, <x>=1.

Syntax

```
>gdi HSC<x>INPUT<y> <n>
```

Where: `<x>` is the number representing the corresponding HSCMP module

`<y>` is the number representing the corresponding external analog input CIN

`<n>` is the voltage input value

Examples

```
>gdi HSC1INPUT1 5.00
```

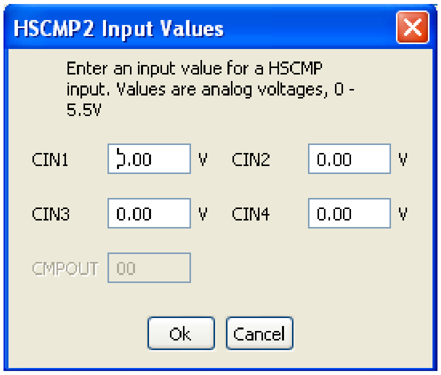
Simulate a 5.00 volt input on CIN1 of HSCMP1.

```
>gdi HSC4INPUT3 2.50
```

Simulate a 2.50 volt input on CIN3 of HSCMP4.

9.1.2.9.3 HSC<x>INPUTS Command

In FCS mode, the HSC<x>INPUTS command opens the HSCMP<x> Input Value dialog box shown in the following figure. The user may then use this box to specify the external analog input voltages. For HCS08 devices that have more than one HSCMP module, <x> is the number representing the corresponding module. For HCS08 devices that only have one HSCMP module, <x>=1.



HSCMP2 Input Values

Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

CIN1 5.00 V CIN2 0.00 V

CIN3 0.00 V CIN4 0.00 V

CMPOUT 00

Figure 9-15. HSCMPx Input Value Dialog Box

When using FCS, the HSC<x>INPUTS command shows the simulated input analog voltages to any applicable HSCMPx module.

Syntax

```
>gdi HSC<x>INPUTS
```

Where:

`<x>` is the number representing the corresponding HSCMP module

Example

```
>gdi HSC1INPUTS
```

Show input analog voltages.

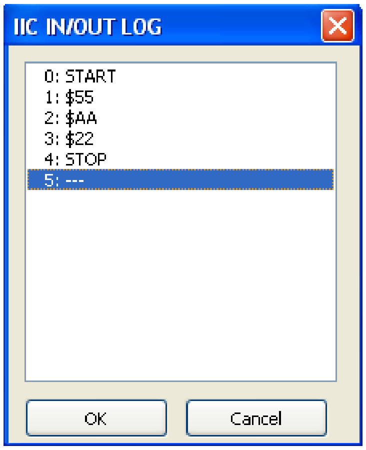
9.1.2.10 Inter-Integrated Circuit Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Inter-Integrated Circuit (IIC) module including:

- Flag polling
- Interrupt enabled mode
- Transmission and reception of external data
- Master and slave modes of operation
- START and STOP signal generation detection
- Acknowledge bit generation detection

FCS mode uses the buffered input/output structure to simulate IIC inputs. You can queue up to 256 data bytes into the input buffer. The output buffer of the USB module can also hold 256 output bytes. To queue the IIC Input Packets, use the `IICDI <...>` command in the command prompt. For a more detailed description of the command, refer to the IIC Commands section. If the IIC packet parameters are properly defined, the packet is placed into the next slot in the input buffer. Otherwise, if no parameters are provided, an IIC Input Buffer window is displayed.

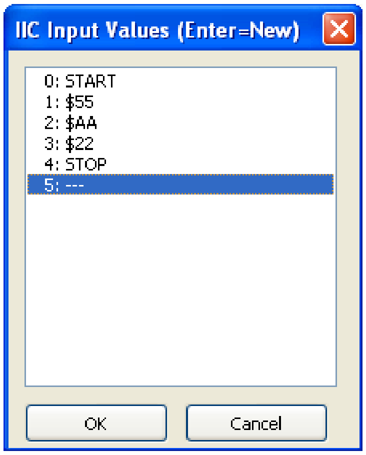
You can enter different IIC packet parameters while the window is open, including START, STOP, ACK, NACK and data bytes. An arrow points to the next byte to be used as input to the IIC. The data from the IIC input buffer is written to the IIC module registers once the IIC module is turned on and properly configured for receiving data from an external IIC device. Once simulation of the data transmission is over, the arrow moves to the next value in the IIC Input Buffer.



```
0: START
1: $55
2: $AA
3: $22
4: STOP
5: ---
```

Figure 9-16. IIC Input Buffer Display

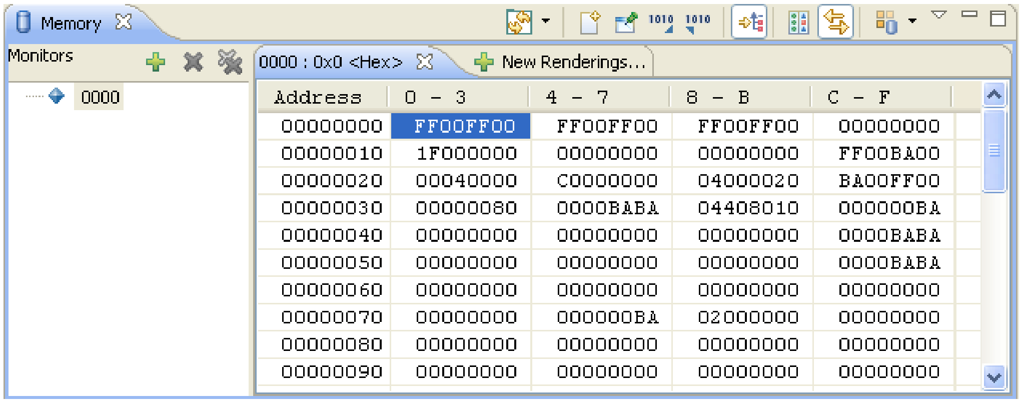
The IIC data input/output log buffer simulation lets you gain access to the past 256 IIC data bytes that have been shifted in and out of the module. To bring up the IIC IN/OUT LOG buffer dialog box, use the `IICDO` command.



```
0: START
1: $55
2: $AA
3: $22
4: STOP
5: ---
```

Figure 9-17. IIC IN/OUT LOG Buffer Display

At any point, use the `IICCLR` command to flush the input as well as input/output log IIC buffers. After the IIC simulated input is received, the first queued-in data byte is passed from the data buffer into the corresponding IIC module registers. It can be observed in the Memory Window by displaying the appropriate register location there.



Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 9-18. Memory Component Window

You can also observe different IIC flags in the Memory window. If you run the module in Flag Polling mode, poll the flag corresponding to the expected IIC event. If the IIC interrupts are enabled, FCS jumps to an appropriate subroutine as long as the IIC interrupt vectors are properly defined.

NOTE

For more information on how to configure IIC module for desired operation, refer to the Freescale user manual for your microprocessor.

9.1.2.10.1 Inter-Integrated Circuit Module Commands

The following commands are available for the HCS08 Inter-Integrated Circuit (IIC) module and the HC08 Multi-Master Inter-Integrated Circuit (MMIIC) module. Command function is identical even though the module names differ.

9.1.2.10.2 IICDI Command

The `IICDI` command lets you input data into a buffer of data to shift into the IIC module when it receives data from an external device. If a data parameter is given, the value is placed into the next slot in the input buffer. Otherwise, if no parameter is given, a window is displayed with the input buffer values. Input values can be entered while the window is open. The maximum number of input values is 256. This command is useful for either inputting response data from a slave target or for inputting data packets from an external master. Note that when the microprocessor attempts to read an acknowledge from an external device, and the next value in the buffer is neither ACK nor NACK, the microprocessor automatically receives an ACK signal (i.e. assumes ACK unless NACK is specified).

Syntax

```
>gdi IICDI [<n>][START][STOP][ACK][NACK]
```

Where: ï¿½ `<n>` indicates the value to be entered into the next location in the input buffer

ï¿½ `START` indicates the incoming START signal

ï¿½ `STOP` indicates the incoming STOP signal

ï¿½ `ACK` corresponds to ACK signal

ï¿½ `NACK` corresponds to NACK signal

NOTE

For a detailed description of the IIC protocol and a proper way to configure the IIC module, refer to the Freescale user manual for your microprocessor.

Example

```
>gdi IICDI
```

Pulls up the data window with all the input values

```
>gdi IICDI 22 33
```

This is an example of data being returned from a slave device. Once the MCU transmits a start signal and the target address, it receives an ACK from the slave device. An ACK is implied unless a NACK is specified via the IICDI command. The next two data bytes read are 22 and 23. If the microprocessor attempts to read another byte, it gets an $FF value followed by a NACK signal (NACK because nothing remains in the input buffer). The receiving device then generates a STOP signal. A more exact input from a device designed to return two bytes is:

```
>gdi IICDI ACK 22 ACK 23 NACK
```

IIC in master mode transmits to a slave:

ï¿½ If the slave device acknowledges all output bytes of the transmitting device, there is no need to specify an input packet. If the master device is going to transmit an address and two bytes, the following packet is equivalent to no packet:

```
>gdi IICDI ACK ACK ACK
```

ï¿½ If, however, the slave receiver is designed to generate a NACK signal after the second received data byte, the proper response packet is:

```
>gdi IICDI ACK ACK NACK
```

ï¿½ The address result being the first ACK, the first data result being the second ACK, and the second data byte being the NACK.

IIC in MASTER mode is not acknowledged by any Slave:

```
>gdi IICDI NACK
```

ï¿½ If the NACK signal is entered before the master device transmits a START signal, then the master device gets a NACK when it tries to read an acknowledge after the address is output. The master device then generates a STOP signal and releases the BUS.

IIC in SLAVE mode receives a Write from an external Master:

This example is for an external master that is writing to the microprocessor configured to simulate the slave mode operation. The packet contains both START and STOP signals which puts the simulated device into the slave mode.

```
>gdi IICDI START 55 AA 22 STOP
```

This input adds five values to the input queue, which is a packet from an external master, including the following procedure values:

ï¿½ A start signal comes in

ï¿½ The address $55 comes in specifying a write (slave receive). The Address Register in the current simulated device has been previously set to $55

ï¿½ The data byte $AA comes in

ï¿½ The data byte $22 comes in

ï¿½ A STOP signal comes in

9.1.2.10.3 IICDO Command

The IICDO command displays a window, which shows data both shifted in and shifted out of the IIC peripheral. An arrow points to the last output value transmitted/received. The maximum number of output values that the buffer can hold is 256.

Syntax

```
>gdi IICDO
```

Example

```
>gdi IICDO
```

View data from the input/output log buffer for IIC simulation.

IICCLR Command Use the `IICCLR` command to flush the input and output buffers for IIC simulation. This resets the buffers and clears all values. Notice that if the IIC is currently shifting a value, this command does not prevent the IIC from finishing the transfer.

Syntax

```
>gdi IICCLR
```

Example

```
>gdi IICCLR
```

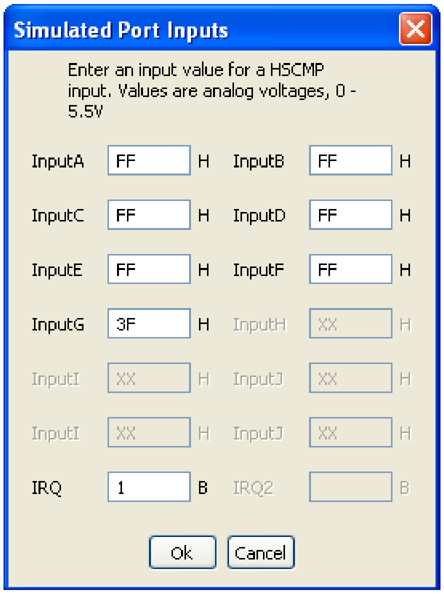
Clear input and output buffers for IIC simulation.

9.1.2.11 Interrupt Priority Controller Module

In Full Chip Simulation (FCS) Mode, this module simulates all the functionality of the Interrupt Priority Controller (IPC) module. This module provides a hardware-based, nested-interrupt mechanism in HCS08 MCUs and allows all prioritized non-software interrupts to interrupt. IPC features a four-level programmable interrupt priority for each source, supports prioritized preemptive interrupt service routines, and the interrupt priority mask can be modified during main flow or interrupt service execution. When the interrupt vector is being fetched, the module can automatically update the interrupt priority mask with its serviced interrupt source priority level and automatically store previous interrupt mask levels.

9.1.2.12 External Interrupt (IRQ) Module

In Full Chip Simulation (FCS) mode, this module simulates the input, flag polling and interrupt functionality of the External Interrupt (IRQ) module. FCS mode uses the `INPUTS` command and let you monitor and change the simulated value of the IRQ input pin state. Once you enter the `INPUTS` command into the command line prompt, the Simulated Port Inputs window appears. Refer the `INPUT<x>` Command for more information about the various forms of this command. In addition, the state of the IRQ pin can be modified directly using the `IRQ<n>` command (documented below).



Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

InputA	FF	H	InputB	FF	H
InputC	FF	H	InputD	FF	H
InputE	FF	H	InputF	FF	H
InputG	3F	H	InputH	XX	H
InputI	XX	H	InputJ	XX	H
InputI	XX	H	InputJ	XX	H
IRQ	1	B	IRQ2		B

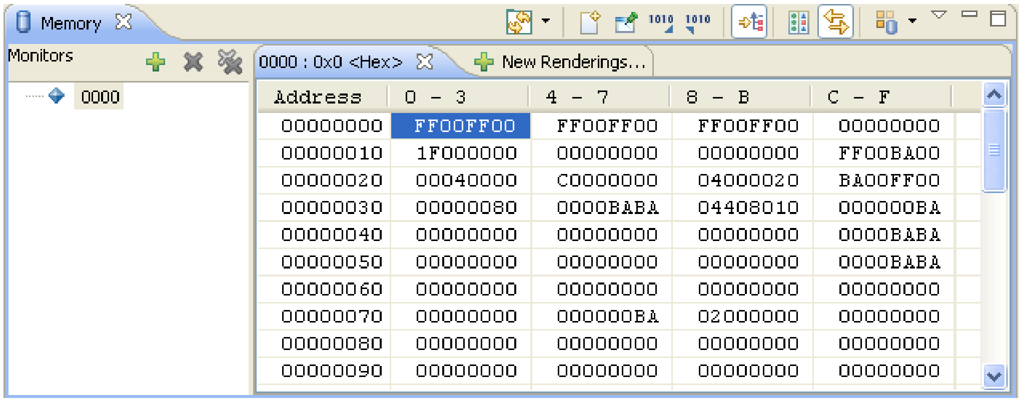
Figure 9-19. Simulated Port Inputs Dialog Box

An IRQ event occurrence sets the appropriate flag in the corresponding IRQ register. You can poll the IRQ flag if the Polling Mode is simulated. In the Interrupt Mode, the simulator branches to an appropriate interrupt subroutine as long as the IRQ interrupt vector is properly configured.

NOTE

For more information on IRQ configuration, refer to the Freescale user manual for your microprocessor.

Following the IRQ event, you can observe the IRQ Flag being set in the IRQ Status and Control register.



```
Address    0 - 3     4 - 7     8 - B     C - F
00000000   FF00FF00  FF00FF00  FF00FF00  00000000
00000010   1F000000  00000000  00000000  FF00BA00
00000020   00040000  C0000000  04000020  BA00FF00
00000030   00000080  0000BABA  04408010  000000BA
00000040   00000000  00000000  00000000  0000BABA
00000050   00000000  00000000  00000000  0000BABA
00000060   00000000  00000000  00000000  00000000
00000070   00000000  000000BA  02000000  00000000
00000080   00000000  00000000  00000000  00000000
00000090   00000000  00000000  00000000  00000000
```

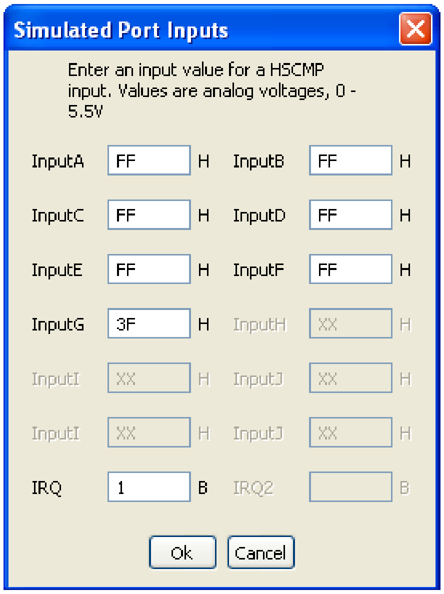
Figure 9-20. Memory Component Window

9.1.2.12.1 IRQ Commands

The following interrupt request command is available for the HC08/HCS08 processors.

9.1.2.12.2 INPUTS Command

In FCS and CPU-Only Simulation mode, the INPUTS command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

InputA	FF	H	InputB	FF	H
InputC	FF	H	InputD	FF	H
InputE	FF	H	InputF	FF	H
InputG	3F	H	InputH	XX	H
InputI	XX	H	InputJ	XX	H
InputI	XX	H	InputJ	XX	H
IRQ	1	B	IRQ2		B

Figure 9-21. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values for any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

Show I/O port input values.

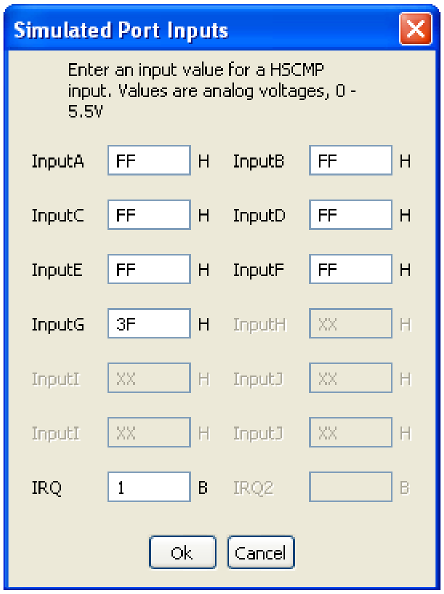
NOTE

The IRQ pin state can be directly manipulated with the `IRQ` command. For example, IRQ 1 simulates a high state on the IRQ pin; likewise, IRQ 0 simulates a logic-low state on the IRQ pin.

9.1.2.13 Keyboard Interrupt Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Keyboard Interrupt (KBI) module, including the edge-only, edge and level interrupt, and flag polling modes of operation. FCS mode uses simulated port inputs to trigger the KBI event from the proper I/O port pin.

To define an input state of the specific port, write the `INPUT<x> <n>` command in the Command window. The `<x>` represents the corresponding I/O port, while `<n>` stands for the input value to write to this port. At the same time, you can use the `INPUTS` command to bring up the Simulated Port Inputs for all general I/O ports. It displays the current simulated values to all applicable input ports. Refer the documentation for Timer Module Commands for more information about the various forms of this command.

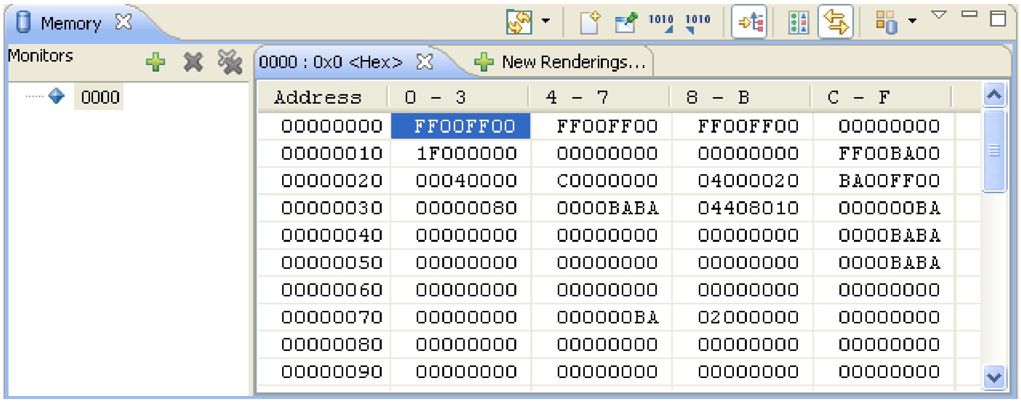


Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

InputA	FF	H	InputB	FF	H
InputC	FF	H	InputD	FF	H
InputE	FF	H	InputF	FF	H
InputG	3F	H	InputH	XX	H
InputI	XX	H	InputJ	XX	H
InputI	XX	H	InputJ	XX	H
IRQ	1	B	IRQ2		B

Figure 9-22. Simulated Port Inputs Dialog Box

Use the Simulated Port Inputs dialog box to reconfigure the input value to any I/O port. To trigger the event, manipulate the inputs to the port in the appropriate manner, depending on whether the KBI is configured for edge-only or edge and level. Once the KBI event takes place, you can observe the KEYF Flag bit, which is a part of the Keyboard Status and Control register, in the Memory window.



Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 9-23. Memory Component Window

You can poll the KBI Interrupt Pending flag if the Polling Mode is simulated. In Interrupt Mode, the simulator branches to an appropriate interrupt subroutine as long as the KBI interrupt vector is properly configured.

NOTE

For more information on KBI configuration, refer to the Freescale user manual for your microprocessor.

9.1.2.13.1 Keyboard Interrupt Commands

Use the following commands for Keyboard Interrupt manipulation.

9.1.2.13.2 INPUT<x> Command

The `INPUT<x>` command sets the simulated inputs to port <x>. The CPU reads this input value when port <x> is set as an input port.

Syntax

```
>gdi INPUT<x> <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` is an eight-bit simulated value for port <x>

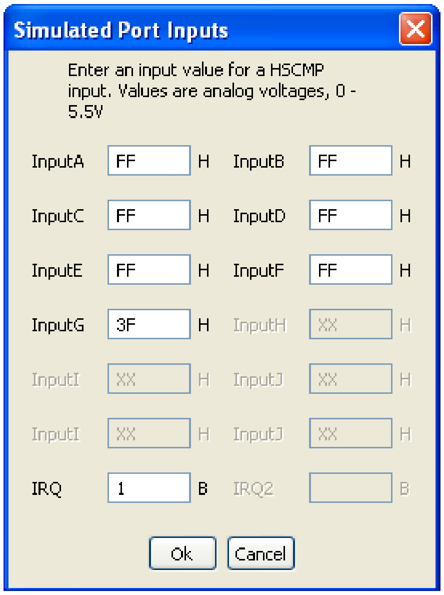
Example

```
>gdi INPUTA AA
```

Simulate the input AA on port A.

9.1.2.13.3 INPUTS Command

In FCS and CPU-Only Simulation mode, the `INPUTS` command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



Simulated Port Inputs

Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

InputA	FF	H	InputB	FF	H
InputC	FF	H	InputD	FF	H
InputE	FF	H	InputF	FF	H
InputG	3F	H	InputH	XX	H
InputI	XX	H	InputJ	XX	H
InputI	XX	H	InputJ	XX	H
IRQ	1	B	IRQ2		B

Figure 9-24. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values to any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

Show I/O port input values.

9.1.2.14 Liquid Crystal Display Driver Module

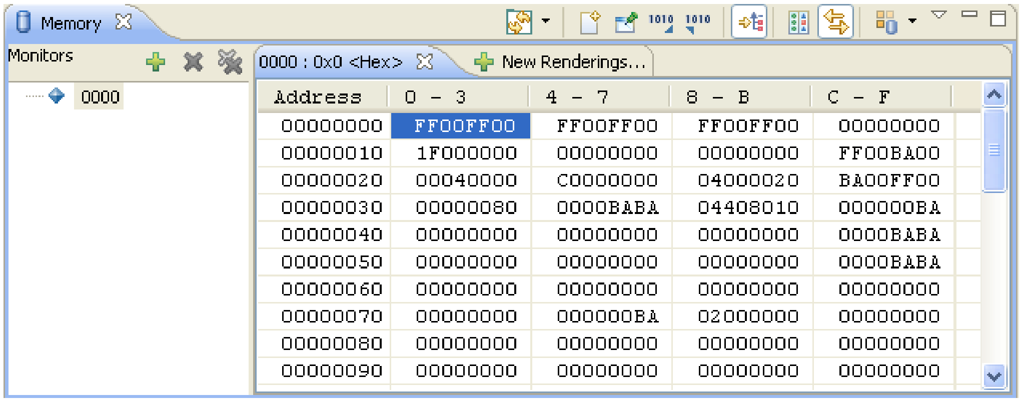
In Full Chip Simulation (FCS) mode, this option lets you simulate all the functionality of the Liquid Crystal Display (LCD) module, including programmable LCD frame frequency, front plane pin configuration, back plane pin configuration, programmable blink frequency, and LCD interrupt flag generation. By default, LCD front and back plane pins are mapped to match device use on the corresponding Freescale DEMO9S08xx device board.

9.1.2.15 Modulo Timer Interrupt Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Modulo Timer Interrupt (MTIM) Module, including:

- Programmable MTIM clock input
- Free running or modulo up count operation
- Flag polling
- Interrupt enabled mode of operation

Once the MTIM Status and Control register properly configures the operation of the module, the MTIM Counter starts incrementing. If modulo up count operation is enabled, you can observe the MTIM overflow flag in the MTIM Status and Control register in the Memory window.



```
Address   0 - 3     4 - 7     8 - B     C - F
00000000  FF00FF00  FF00FF00  FF00FF00  00000000
00000010  1F000000  00000000  00000000  FF00BA00
00000020  00040000  C0000000  04000020  BA00FF00
00000030  00000080  0000BABA  04408010  000000BA
00000040  00000000  00000000  00000000  0000BABA
00000050  00000000  00000000  00000000  0000BABA
00000060  00000000  00000000  00000000  00000000
00000070  00000000  000000BA  02000000  00000000
00000080  00000000  00000000  00000000  00000000
00000090  00000000  00000000  00000000  00000000
```

Figure 9-25. Memory Component Window

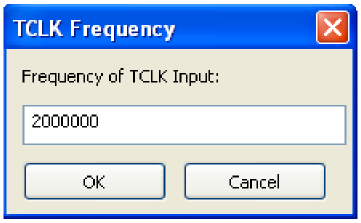
If the MTIM interrupt is enabled, the FCS jumps to an appropriate subroutine as long as the MTIM interrupt vector is properly defined.

9.1.2.15.1 Modulo Timer Interrupt Module User Commands

The following commands are available for the MTIM.

9.1.2.15.2 TclK Command

The TclK opens the TclK frequency dialog box shown in the following figure. You may then use this box to specify the input frequency of the TclK.



Frequency of TCLK Input: 2000000

Figure 9-26. TclK Frequency Dialog Box

Syntax

```
>gdi TclK
```

Example

```
>gdi TclK
```

Show TclK Frequency Dialog Box

9.1.2.15.3 TclK <n> Command

The TclK <n> command sets the TclK input frequency to <n>.

Syntax

```
>gdi TclK <n>
```

Where: `<n>` is the input frequency of TclK

Example

```
>gdi TclK 200000t
```

Simulate the TclK input frequency of 200000 Hz.

9.1.2.16 MSCAN Controller Module

The MSCAN Controller Module fully simulates the operation of the MSCAN08 Protocol Version 2.0 based device, including:

- Flag polling
- Interrupt enabled mode
- 0-8 bytes data length
- Transmission and reception of external data

The MSCAN08 peripheral is a scalable Control Area Network (CAN) 2.0 compliant device that allows microcontrollers to exchange data between themselves at high speeds. This is done through a high-speed serial link that is deterministic and reliable. CAN devices are often utilized in automobiles, where multiple microcontrollers need to be connected into a network. The CAN specification indicates that any unit on the bus can be a master at any time, which sends a message to another unit at any time, provided the bus is free to do so. All of these messages can be set up through the CAN I/O commands built into the simulator. This section goes through an example which shows how the simulator can be used to test out code that drives the CAN peripheral

9.1.2.17 Programmable Delay Block Module

In Full Chip Simulation (FCS) Mode, this module lets you simulate all the functionality of the Programmable Delay Block (PDB) module. This module's primary function is to provide a controllable delay from FTM SYNC output to the sample trigger input of PGA or ADC, or a controllable window synchronized with PWM pulses for ACMP to compare the analog signals in a defined window. PDB can alternately generate PWM pulses that are synchronized to FTM, CMPR output, and RTC.

9.1.2.18 Programmable Gain Amplifier Module

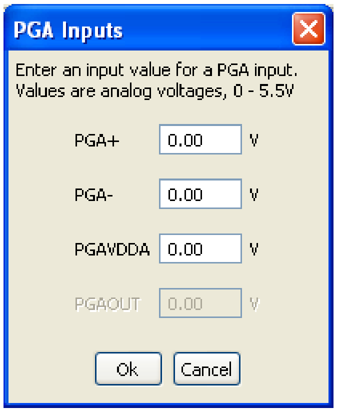
In Full Chip Simulation (FCS) Mode, this option lets you simulate all the functionality of the Programmable Gain Amplifier (PGAV1) module, including data input on all PGA channels, flag polling, and output connection to the ADC input channel. The user can utilize either the PGA Inputs display form or command-line commands to provide inputs to the PGA module. The PGAINPUTS command shows the simulated PGA input/output values. There are also three specific commands in the simulation for providing PGA inputs to simulation via a command line. They are: PGAINPLUS <x>, PGAINMINUS <x>, PGAINVDDA <x>. These commands allow the user to automate the testing/debugging procedure without relying on a modal form for entering the data.

9.1.2.18.1 Programmable Gain Amplifier User Commands

The following commands are available for the HCS08 PGAV1 Module.

9.1.2.18.2 PGAINPUTS Command

The PGAINPUTS command shows the simulated PGA input/output values. The user may use this window to specify the input values of PGA+, PGA-, and PGAVDDA. The read-only PGAOUT is the value of PGA output.



Enter an input value for a PGA input.
Values are analog voltages, 0 - 5.5V

PGA+ 0.00 V

PGA- 0.00 V

PGAVDDA 0.00 V

PGAOUT 0.00 V

Figure 9-27. PGA Inputs Window

The PGAINPUTS command can be used at any point to bring up this PGA Inputs window. The window displays the input values of PGA+, PGA-, PGAVDDA and the output value of PGAOUT.

Syntax

```
>gdi PGAINPUTS
```

Examples

```
>gdi PGAINPUTS
```

Pull up the PGA Inputs window

9.1.2.18.3 PGAINPLUS <x> Command

The PGAINPLUS <x> command allows user to input a value for PGA+ channel.

Syntax

```
>gdi PGAINPLUS <x>
```

Where: `<x>` is the input value in volts

Examples

```
>gdi PGAINPLUS 2.00
```

Simulate the input 2.00 volts for the PGA+ channel

9.1.2.18.4 PGAINMINUS <x> Command

The PGAINMINUS <x> command allows the user to input a value for the PGA- channel.

Syntax

```
>gdi PGAINMINUS <x>
```

Where: `<x>` is the input value in volts

Examples

```
>gdi PGAINMINUS 3.50
```

Simulate the input 3.50 volts for PGA- channel

9.1.2.18.5 PGAINVDDA <x> Command

The PGAINVDDA <x> command allows the user to input a value for the VDDA signal that the PGA module connects to.

Syntax

```
>gdi PGAINVDDA <x>
```

Where: `<x>` is the input value in volts

Examples

```
>gdi PGAINVDDA 1.50
```

Simulate the input 1.50 volts for VDDA signal

9.1.2.19 Programmable Reference Analog Comparator Module

In Full Chip Simulation (FCS) Mode, this option lets you simulate all the functionality of the Programmable Reference Analog Comparator (PRACMP) module, including data input on all PRACMP channels, flag polling, and interrupt operation. The user can use either the PRACMP Inputs display form or command-line commands to provide inputs to PRACMP module.

9.1.2.19.1 Programmable Reference Analog Comparator User Commands

The following commands are available for the HCS08 PRACMP Module.

9.1.2.19.2 PR<x>INPUT<y> Command

The PR<x>INPUT<y> command allows the user to input voltage value for the external analog input CIN<y>. For HCS08 devices that have more than one PRACMP module, <x> is the number representing the corresponding module. For HCS08 devices that only have one PRACMP module, <x>=1.

Syntax

```
>gdi PR<x>INPUT<y> <n>
```

Where: `<x>` is the number representing the corresponding PRACMP module

`<y>` is the number representing the corresponding external analog input CIN

`<n>` is the voltage input value

Examples

```
>gdi PR1INPUT1 5.00
```

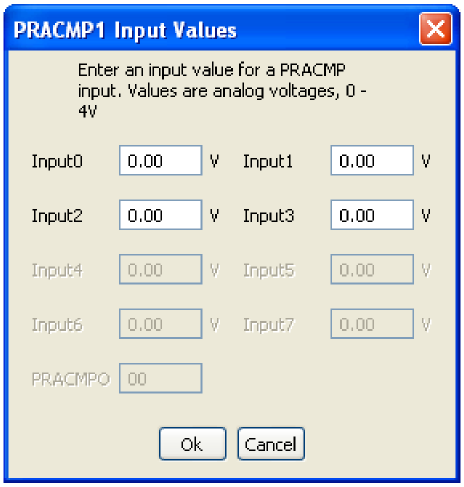
Simulate the input 5.00 volts on CIN1 of PRACMP1.

```
>gdi PR4INPUT3 2.50
```

Simulate the input 2.50 volts on CIN3 of PRACMP4.

9.1.2.19.3 PR<x>INPUTS Command

In FCS Mode, the PR<x>INPUTS command opens the PRACMP<x> Input Value dialog box shown in the following figure. The user may then use this box to specify the external analog input voltages. For HCS08 devices that have more than one PRACMP module, <x> is the number representing the corresponding module. For HCS08 devices that only have one PRACMP module, <x>=1.



Enter an input value for a PRACMP input. Values are analog voltages, 0 - 4V

Input0 0.00 V Input1 0.00 V
Input2 0.00 V Input3 0.00 V
Input4 0.00 V Input5 0.00 V
Input6 0.00 V Input7 0.00 V
PRACMPO 00

Figure 9-28. PRACMPx Input Value dialog box

When using FCS, the PR<x>INPUTS command shows the simulated input analog voltages to any applicable PRACMPx module.

Syntax

```
>gdi PR<x>INPUTS
```

Where: `<x>` is the number representing the corresponding PRACMP module

Example

```
>gdi PR1INPUTS
```

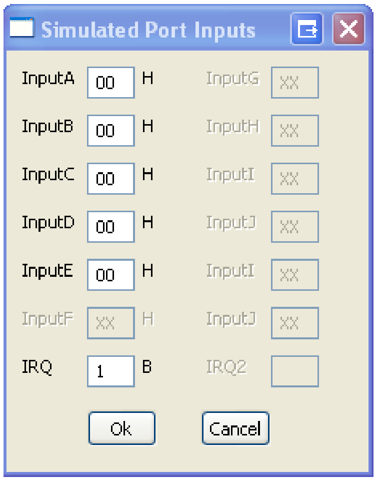
Show input analog voltages of PRACMP1.

9.1.2.20 Input/Output (I/O) Ports Module

In Full Chip Simulation (FCS) mode, this module simulates all input and output functionality of the Input/Output (I/O) Ports module. FCS mode uses a set of designated commands to simulate the input and output activity on corresponding I/O port pins. To define an input state of a specific port, write the INPUT <x> <n> command in the Command window. The <x> represents the corresponding I/O port, while the <n> stands for the input value to write to this port. At the same time, you can use the INPUTS command to bring up the Simulated Port Inputs for all general I/O ports. It displays the current simulated values to all applicable input ports.

NOTE

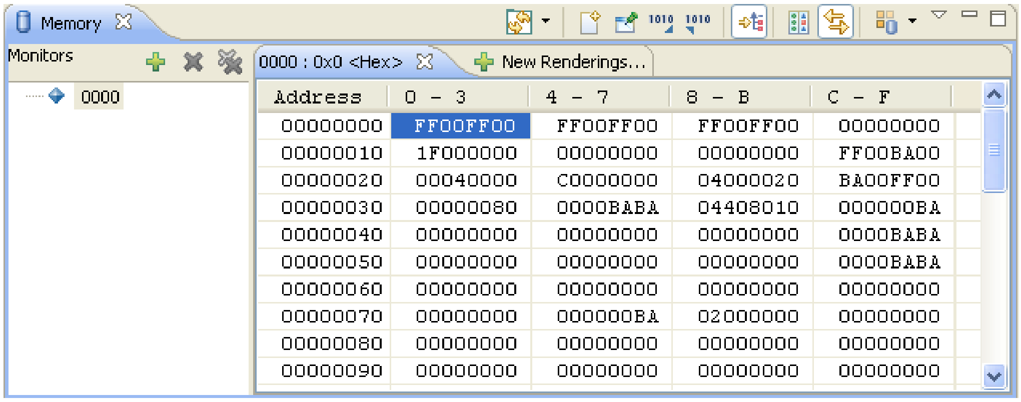
> Refer the Input/Output Ports User Commands and IRQ Commands for more information about the various forms of this command.



InputA	00	H	InputG	XX
InputB	00	H	InputH	XX
InputC	00	H	InputI	XX
InputD	00	H	InputJ	XX
InputE	00	H	InputI	XX
InputF	XX	H	InputJ	XX
IRQ	1	B	IRQ2	

Figure 9-29. Simulated Port Inputs Dialog Box

Use the Simulated Port Inputs dialog box to reconfigure the input value to any I/O port. Use the INPUTS command to reconfigure the output values on any relevant I/O port. You can observe the manipulation of I/O port pins in the Memory window.



Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 9-30. Memory Component Window

Note that if the regular I/O pins are multiplexed to be used by a different MCU Module, they might not be available for general I/O functionality.

NOTE

For more information on how to properly configure I/O pins, refer to the Freescale user manual for your microprocessor.

9.1.2.20.1 Input/Output Ports User Commands

Use the following commands for general I/O ports manipulation.

9.1.2.20.2 INPUT<x> Command

The INPUT<x> command sets the simulated inputs to port <x>. The CPU reads this input value when port <x> is set as an input port.

Syntax

```
>gdi INPUT<x> <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` Eight-bit simulated value for port <x>

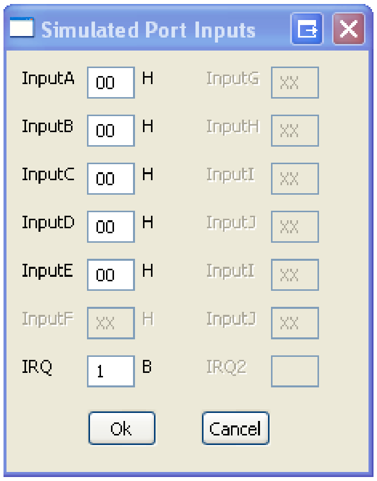
Example

```
>gdi INPUTA AA
```

Simulate the input AA on port A.

9.1.2.20.3 INPUTS Command

In FCS and CPU-Only Simulation mode, the INPUTS command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



InputA 00 H   InputG XX
InputB 00 H   InputH XX
InputC 00 H   InputI XX
InputD 00 H   InputJ XX
InputE 00 H   InputI XX
InputF XX H   InputJ XX
IRQ 1 B   IRQ2

Figure 9-31. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values to any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

Show I/O port input values.

9.1.2.20.4 DDR<x>OUT Command

For devices that have a dedicated enable register for input and output GPIO functionality, the DDR<x>OUT command sets the simulated pins on port <x> as an output and clears corresponding input enable bit (see DDR<x>IN). Use the INPUT <x> command to change the values of the port pins.

Syntax

```
>gdi DDR<x>OUT <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` Eight-bit simulated value for port <x>

Example

```
>gdi DDRBOUT F0
```

Simulate port B with bits 4-7 as output.

9.1.2.20.5 DDR<x>IN Command

For devices that have a dedicated enable register for input and output GPIO functionality, the DDR<x>IN command sets the simulated pins on port <x> as an input and clears corresponding output enable bit (see DDR<x>OUT). Use the INPUT <x> command to change the values of the port pins.

Syntax

```
>gdi DDR<x>IN<n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` Eight-bit simulated value for port <x>

Example

```
>gdi DDRCIN 0F
```

Simulate port C with bits 0-3 as input.

9.1.2.21 Serial Communications Interface Module

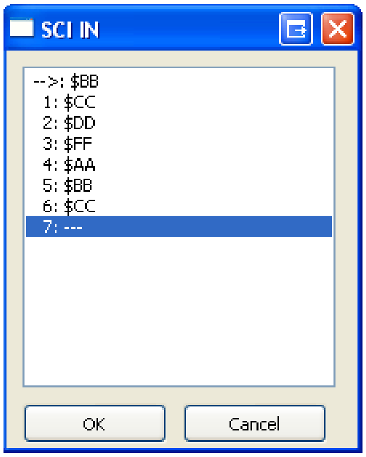
In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Serial Peripheral Interface (SPI) module including:

- Flag polling
- Interrupt enabled mode
- 8- or 9-bit length data codes
- Odd and even parity modes
- Transmission and reception of external data

FCS mode uses the buffered input/output structure to simulate SCI inputs. You can queue up to 256 data values into the input buffer. The output buffer of the SCI module can also hold 256 output values. To queue the SCI Input Data, use the SCDI <n> command in the command prompt. If <n> (the data parameter) is given, the value is placed into the next slot in the input buffer.

Otherwise, if no parameter is provided, a window is displayed with the input buffer values. You can enter input values while the window is open. An arrow points to the next value to be used as input to the SCI. The data from the SCI input buffer is written to the

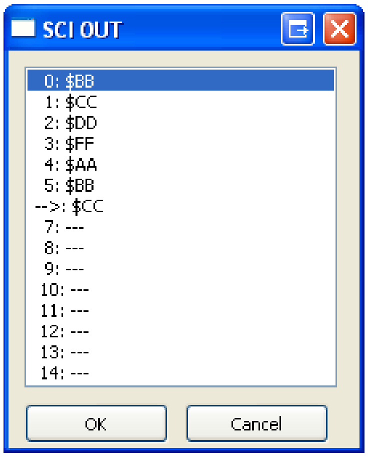
SCI data register once the SCI module has been turned on and is properly configured for receiving data from an external serial device. Once the simulation of the data transmission is over, the arrow moves to the next value in the SCI IN Buffer.



```
-->: $BB
 1: $CC
 2: $DD
 3: $FF
 4: $AA
 5: $BB
 6: $CC
 7: ---
```

Figure 9-32. SCI IN Buffer Display

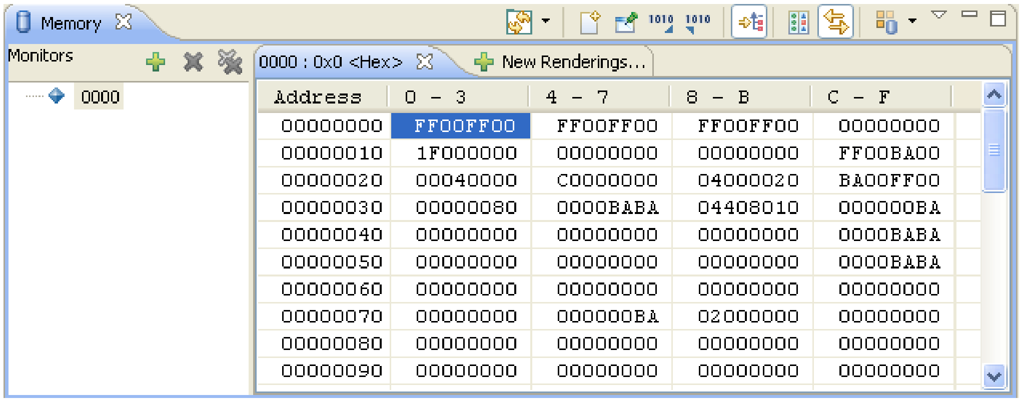
SCI Data Output Buffer simulation lets you gain access to the past 256 SCI data values transmitted out of the module. To bring up the SCI OUT buffer dialog box, use the SCDO command.



```
 0: $BB
 1: $CC
 2: $DD
 3: $FF
 4: $AA
 5: $BB
-->: $CC
 7: ---
 8: ---
 9: ---
10: ---
11: ---
12: ---
13: ---
14: ---
```

Figure 9-33. SCI OUT Buffer Display

The SCCLR command may be used at any point to flush the input and output SCI buffers. After the SCI simulated input is received, the first queued value is passed from the data buffer into the SCI data register. It can be observed in the memory window by displaying the memory location corresponding to the SCI data register.



```
Address   0 - 3     4 - 7     8 - B     C - F
00000000  FF00FF00  FF00FF00  FF00FF00  00000000
00000010  1F000000  00000000  00000000  FF00BA00
00000020  00040000  C0000000  04000020  BA00FF00
00000030  00000080  0000BABA  04408010  000000BA
00000040  00000000  00000000  00000000  0000BABA
00000050  00000000  00000000  00000000  0000BABA
00000060  00000000  00000000  00000000  00000000
00000070  00000000  000000BA  02000000  00000000
00000080  00000000  00000000  00000000  00000000
00000090  00000000  00000000  00000000  00000000
```

Figure 9-34. Memory Component Window

You can also observe different SCI flags in the Memory window. If the module is run in Flag Polling mode, poll the flag corresponding to the expected SCI event. If the SCI interrupts are enabled, the FCS jumps to an appropriate subroutine as long as the SCI interrupt vectors are properly defined.

NOTE

For more information on how to configure the SCI module for desired operation, refer to the Freescale user manual for your microprocessor.

9.1.2.21.1 SCI Commands

The following serial communication interface commands are available for the HC08/ HCS08.

9.1.2.21.2 SCCLR Command

Use the SCCLR command to flush the input and output buffers for SCI simulation. This resets the buffers and clears out all values. Note that if the SCI is in the process of shifting a value, this command allows the SCI to finish the transfer. Refer the SCDI and SCDO commands for accessing the input and output buffers of the SCI interface.

Syntax

```
>gdi SCCLR
```

Example

```
>gdi SCCLR
```

Clear input and output buffer for SCI simulation

9.1.2.21.3 SCDI Command

The SCDI command lets you input data into the SCI. If a data parameter is given, the value is placed into the next slot in the SCI input buffer. If no parameter is given, a window displays the input buffer values. Input values can be entered while the window is open. An arrow points to the next value to be used as input to the SCI. The maximum number of input values is 256 bytes.

Syntax

```
>gdi SCDI [<n>]
```

Where: `<n>` The value to be entered into the next location in the input buffer

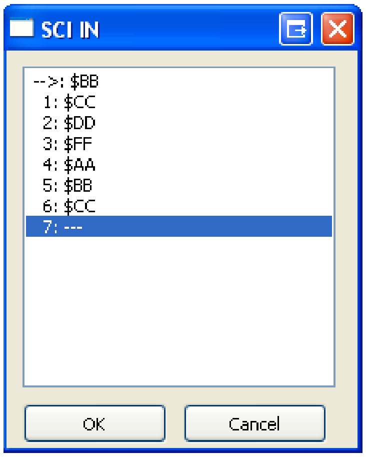
Example

```
>gdi SCDI $55
```

Set the next input value to the SCI to $55

```
>gdi SCDI
```

Pull up the data window with all the input values.



```
-->: $BB
  1: $CC
  2: $DD
  3: $FF
  4: $AA
  5: $BB
  6: $CC
  7: ---
```

Figure 9-35. SCI IN Buffer Display

9.1.2.21.4 SCDO Command

The `SCDO` command displays the output buffer from the SCI. A window is opened that shows all the data that the SCI has shifted out. An arrow points to the last output value transmitted. The maximum number of output values that the buffer holds is 256 bytes.

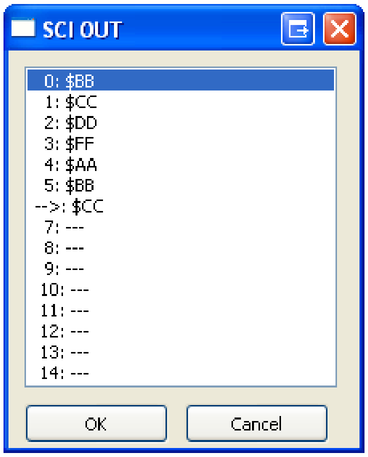
Syntax

```
>gdi SCDO
```

Example

```
>gdi SCDO
```

View data from the output buffer for the SCI simulation.



```
0: $BB
1: $CC
2: $DD
3: $FF
4: $AA
5: $BB
-->: $CC
7: ---
8: ---
9: ---
10: ---
11: ---
12: ---
13: ---
14: ---
```

Figure 9-36. SCI OUT Buffer Display

9.1.2.22 Slave LIN Interface Controller (SLIC) Module

In Full Chip Simulation (FCS) mode, this option will simulate all functionality of the Slave LIN Interface Controller (SLIC) Module, including:

- Flag polling
- Interrupt enabled mode
- Transmission and reception of external data
- Check sum generation and verification
- Different message lengths data modes

Full Chip Simulation (FCS) mode uses a buffered structure to simulate SLIC inputs and outputs. You can queue up to 256 data bytes into the input buffer. The output buffer of the SLIC module can also hold 256 output bytes. To queue the SLIC Input bytes use the SLCIN instruction in the command prompt. For a more detailed description of the command, refer to the SLIC Commands section. The SLIC instruction brings up a window, which displays a list of queued input data. Different SLIC packets can be entered while the window is open. An arrow points to the byte that will be used next as input to the SLIC. Once the SLIC module is turned on and properly configured for receiving data from an external SLIC device, the data from the SLIC input buffer is written to the SLIC module identifier or data registers. After the simulation of the data transmission is complete, the arrow moves to the next value in the SLIC IN Buffer.

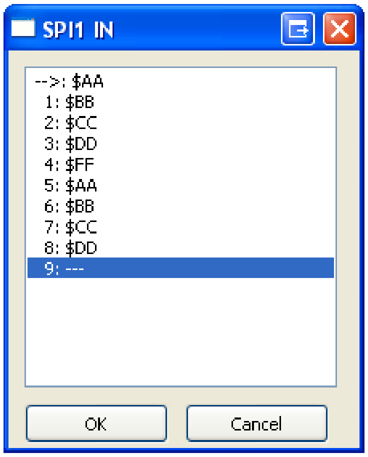
9.1.2.23 Serial Peripheral Interface Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Serial Peripheral Interface (SPI) module including:

- Flag polling
- Interrupt enabled mode
- Master and slave modes
- Slave input clock
- Transmission and reception of external data

FCS mode uses the buffered input/output structure to simulate SPI inputs. You can queue up to 256 data values into the input buffer. The output buffer of the SPI module can also hold 256 output values. To queue the SPI Input Data, use the SPDI <n> command at the command prompt. If <n> (the data parameter) is given, the value is placed into the next slot in the input buffer.

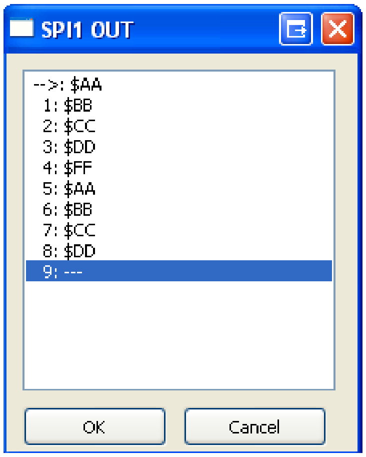
Otherwise a window is displayed with the input buffer values. You can enter input values while the window is open. An arrow points to the next input value to the SPI. The data from the SPI input buffer is written to the SPI data register once the SPI module is turned on and is properly configured for receiving data from an external serial device. Once the simulation of the data transmission is over, the arrow moves to the next value in the SPI IN Buffer.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 9-37. SPI IN Buffer Display

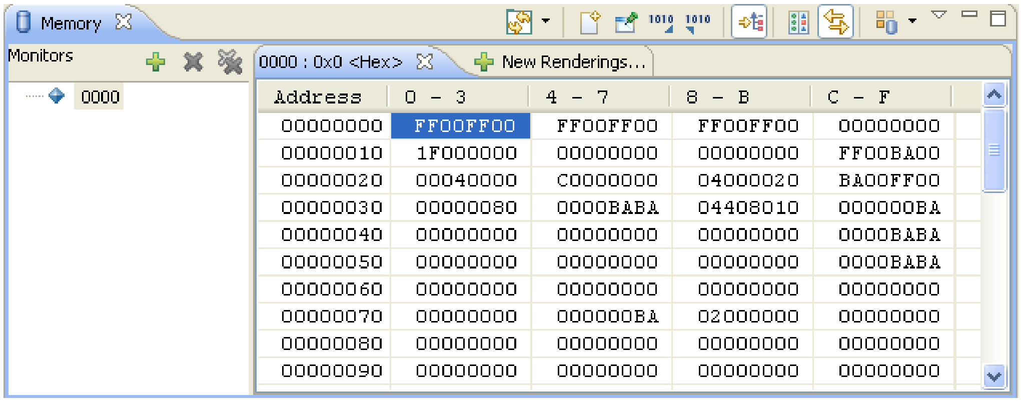
SPI data output buffer simulation lets you gain access to the past 256 SPI data values transmitted out of the module. To bring up the SPI OUT buffer dialog box, use the SPDO command.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 9-38. SPI OUT Buffer Display

The SPCLR command may be used at any point to flush the input and output SPI buffers. After the SPI simulated input is received, the first queued value is passed from the data buffer into the SPI data register. It can be observed in the Memory Window by displaying the memory location corresponding to the SPI data register.



```
Address     0 - 3      4 - 7      8 - B      C - F
00000000    FF00FF00   FF00FF00   FF00FF00   00000000
00000010    1F000000   00000000   00000000   FF00BA00
00000020    00040000   C0000000   04000020   BA00FF00
00000030    00000080   0000BABA   04408010   000000BA
00000040    00000000   00000000   00000000   0000BABA
00000050    00000000   00000000   00000000   0000BABA
00000060    00000000   00000000   00000000   00000000
00000070    00000000   000000BA   02000000   00000000
00000080    00000000   00000000   00000000   00000000
00000090    00000000   00000000   00000000   00000000
```

Figure 9-39. Memory Component Window

You can also observe different SPI flags in the Memory window. If the module is run in Flag Polling mode, poll the flag corresponding to the expected SPI event. If the SPI interrupts are enabled, the FCS jumps to an appropriate subroutine as long as the SPI channel interrupt vectors are properly defined.

To simulate the frequency of the SPI slave input clock, use the SPFREQ <n> command. If the SPI is configured for slave mode, this command let you enter the number of cycles <n> in the period of the input clock. If the SPFREQ command is not used, then clocking is set by the SPI control register.

NOTE

For more information on how to configure the SPI module for desired operation, refer to the Freescale user manual for your microprocessor.

9.1.2.23.1 SPI Commands

The following serial peripheral interface commands are available for the HCS08.

9.1.2.23.2 SPCLR Command

Use the SPCLR command to flush the input and output buffers for SPI simulation. This resets the buffers and clears out all values. Notice that if the SPI is currently shifting a value, this command allows the SPI to finish the transfer. Refer the SPDI and SPDO commands for accessing the input and output buffers of the SPI interface.

Syntax

```
>gdi SPCLR
```

Example

```
>gdi SPCLR
```

Clear input and output buffer for SPI simulation

9.1.2.23.3 SPDI Command

The SPDI command lets you input data into the SPI. If a data parameter is given, the value is placed into the next slot in the SPI input buffer. If no parameter is given, a window displays the input buffer values. You can enter input values while the window is open. An arrow points to the next input value to the SPI. The maximum number of input values is 256 bytes.

Syntax

```
>gdi SPDI [<n>]
```

Where: `<n>` The value to be entered into the next location in the input buffer

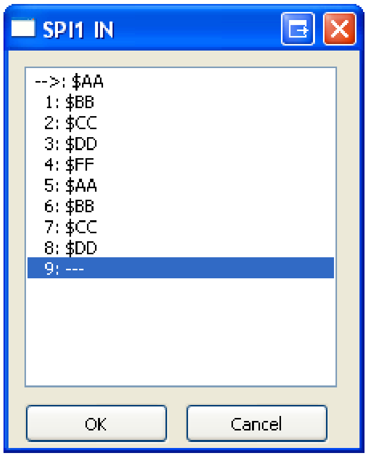
Example

```
>gdi SPDI $55
```

Set the next input value to the SPI to $55

```
>gdi SPDI
```

Pull up the data window with all the input values.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 9-40. SPI IN Buffer Display

9.1.2.23.4 SPDO Command

The SPDO command displays the output buffer from the SPI. A window opens that shows all the data that the SPI has shifted out. An arrow points to the last output value transmitted. The maximum number of output values that the buffer holds is 256 bytes.

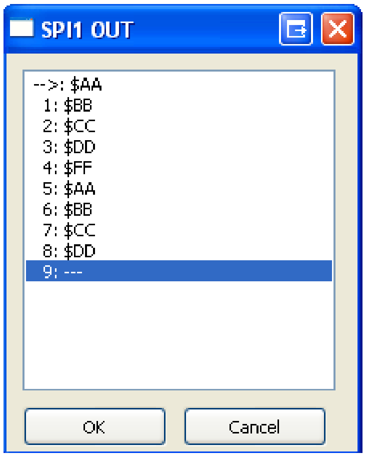
Syntax

```
>gdi SPDO
```

Example

```
>gdi SPDO
```

View data from the output buffer for the SPI simulation.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 9-41. SPI OUT Buffer Display

9.1.2.23.5 SPFREQ Command

The SPFREQ command lets you set the frequency of the SPI slave input clock. If the SPI is configured for the slave mode, this command lets you enter the number of cycles <n> per one input clock period. If no value is given, a window appears and you are prompted for a value. If this command is not used, then the clocking is assumed to be set by the SPI control register.

Syntax

```
>gdi SPFREQ [<n>]
```

Where: `<n>` The number of cycles for the period of the input clock.

Example

```
>gdi SPFREQ 8
```

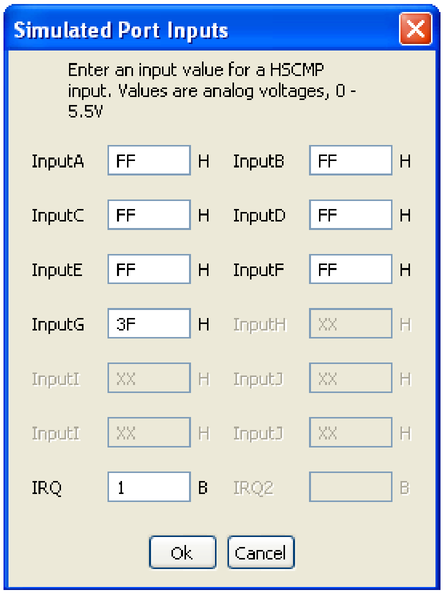
Set the period of the input slave clock to 8 cycles (total shift = 8*8 cycles per bit =64 cycles)

9.1.2.24 Timer Interface Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Timer Interface module, including:

- Input capture/output compare
- Pulse width modulation
- Internal or external clock input
- Free running or modulo up count operation
- Flag polling
- Interrupt enabled mode of operation.

FCS mode uses the simulated port inputs to trigger the input capture on a given timer channel. To define an input state of the specific port, use the INPUT<x> <n> command in the Command window. The <x> represents the corresponding I/O port, while <n> stands for the input value to write to this port. At the same time, the INPUTS command can be used to display the Simulated Port Inputs for all general I/O ports. It displays the current simulated values to all applicable input ports. Refer the documentation for Timer Module Commands for more information about the various forms of this command.



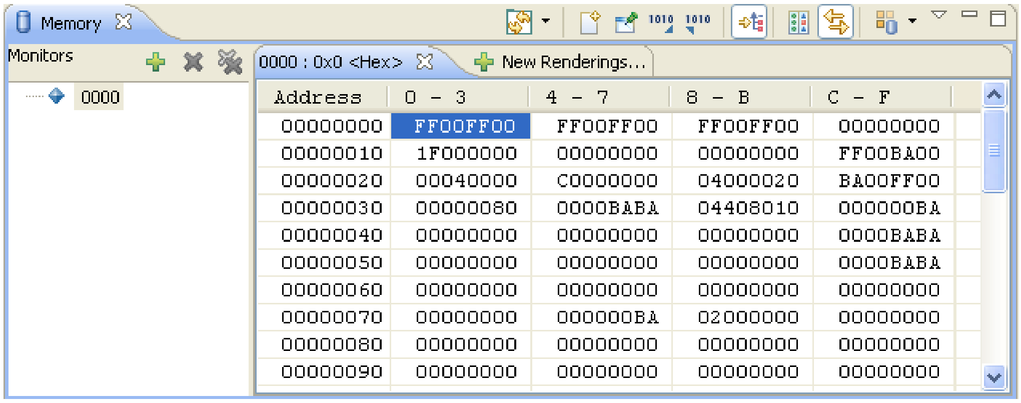
Simulated Port Inputs

Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

InputA	FF	H	InputB	FF	H
InputC	FF	H	InputD	FF	H
InputE	FF	H	InputF	FF	H
InputG	3F	H	InputH	XX	H
InputI	XX	H	InputJ	XX	H
InputI	XX	H	InputJ	XX	H
IRQ	1	B	IRQ2		B

Figure 9-42. Simulated Port Inputs Dialog Box

Use the Simulated Port Inputs dialog box to reconfigure the input value to any I/O port. To trigger the event, first set the port inputs high or low and then invert them to an opposite value, depending on whether the input capture is set for rising/falling edge. Once the Input Capture event takes place you can observe the CHxF in the Channel Status and Control register in the Memory window.



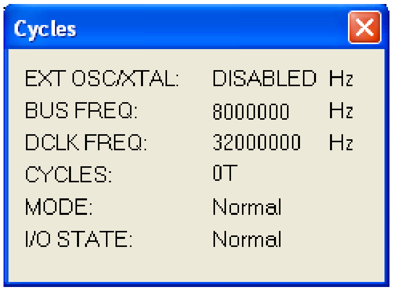
```
Address    0 - 3     4 - 7     8 - B     C - F
00000000   FF00FF00  FF00FF00  FF00FF00  00000000
00000010   1F000000  00000000  00000000  FF00BA00
00000020   00040000  C0000000  04000020  BA00FF00
00000030   00000080  0000BABA  04408010  000000BA
00000040   00000000  00000000  00000000  0000BABA
00000050   00000000  00000000  00000000  0000BABA
00000060   00000000  00000000  00000000  00000000
00000070   00000000  000000BA  02000000  00000000
00000080   00000000  00000000  00000000  00000000
00000090   00000000  00000000  00000000  00000000
```

Figure 9-43. Memory Component Window

If the Timer module is configured for an Output Compare event, then once the event takes place you can observe the same CHxF Flag via the Memory window. If the timer channel interrupt is enabled, the FCS jumps to an appropriate subroutine as long as the Timer channel interrupt vector is properly defined. To observe the Timer Overflow Flag (TOF) flag being set as a result of the corresponding CPU event, situate your Memory window on the memory location of the Timer Status and Control register.

To observe the Pulse Width Modulation (PWM) operation, properly configure the Timer to operate in the Modulo up count mode, select the toggle-on-overflow or clear/set output on compare events to create the desired duty cycle wave. Once a PWM event takes place, you can observe pin toggle/clear/set behavior corresponding to the Timer configuration in the Memory window displaying the I/O port associated with a given timer channel.

To observe the accuracy of the Timer module operation, you can observe the number of CPU cycles that it takes for the event to occur. The cycle counter is only incremented as the you step through the code. To determine the exact amount of cycles over which the event occurs, one can either observe the cycle display in the Register window or use the built in simulation commands. To display the current number of cycles in the Command window, use the CYCLES command. To change the number of cycles in the cycle counter, use CYCLES <n>, where <n> is the new cycle value. If the event has a pre-calculated number of cycles, use CYCLE 00 to reset the number of cycles and GOTOCYCLE <n> to run through the code until you reach the expected event.



EXT OSC/XTAL:	DISABLED	Hz
BUS FREQ:	8000000	Hz
DCLK FREQ:	32000000	Hz
CYCLES:	0T	
MODE:	Normal	
I/O STATE:	Normal	

Figure 9-44. Register Window With Cycles Display

9.1.2.24.1 Timer Module Commands

The following timer module commands are available for use with the HC08/HCS08 processors.

9.1.2.24.2 CYCLES Command

The CYCLES command changes the value of the cycles counter. The cycles counter counts the number of the processor cycles that have passed during execution. The Cycles window shows the cycle counter. The cycle count can be useful for timing procedures.

Syntax

```
>gdi CYCLES <n>
```

Where: `<n>` Integer value for the cycles counter

Examples

```
>gdi CYCLES 0
```

Reset cycles counter

```
>gdi CYCLES 1000
```

Set cycle counter to 1000.

9.1.2.24.3 GOTOCYCLE Command

The GOTOCYCLE command executes the program in the simulator beginning at the address in the program counter (PC). Execution continues until the cycle counter is equal to or greater than the specified value, until a key or the Stop button on the toolbar is pressed, until it reaches a break point, or until an error occurs.

Syntax

```
>gdi GOTOCYCLE <n>
```

Where: `<n>` Cycle-counter value at which the execution stops

Example

```
>gdi GOTOCYCLE 100
```

Execute the program until the cycle counter equals 100.

9.1.2.24.4 INPUT<x> Command

The INPUT<x> command sets the simulated inputs to port <x>. The CPU reads this input value when port <x> is set as an input port.

Syntax

```
>gdi INPUT<x> <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` Eight-bit simulated value for port <x>

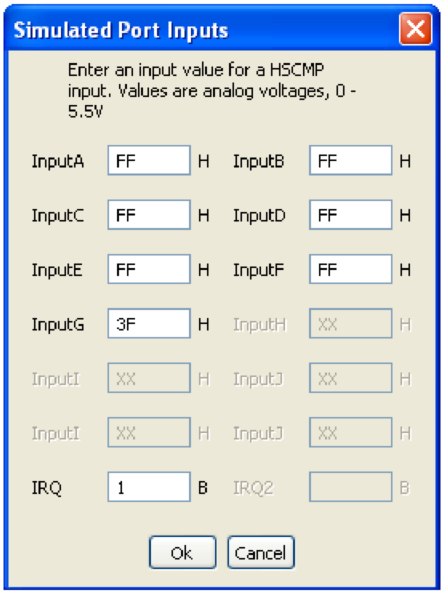
Example

```
>gdi INPUTA AA
```

Simulate the input AA on port A.

9.1.2.24.5 INPUTS Command

In FCS and CPU-Only Simulation mode, the INPUTS command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



Simulated Port Inputs

Enter an input value for a HSCMP input. Values are analog voltages, 0 - 5.5V

InputA	FF	H	InputB	FF	H
InputC	FF	H	InputD	FF	H
InputE	FF	H	InputF	FF	H
InputG	3F	H	InputH	XX	H
InputI	XX	H	InputJ	XX	H
InputI	XX	H	InputJ	XX	H
IRQ	1	B	IRQ2		B

Figure 9-45. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values to any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

Show I/O port input values.

9.1.2.25 Time Of Day Module Option

In Full Chip Simulation (FCS) mode, this module lets you simulate all the functionality of the Time Of Day (TOD) module. The module includes an 8-bit counter, a 6-bit match register, several binary-based and decimal-based prescaler dividers, three clock source options, and one interrupt that can be used for quarter second, one second and match conditions. A 4 Hz signal is used as the reference clock for the TOD counter, where each tick of the TOD counter is 0.25 seconds. This module can be used for time-of-day, calendar, or any task scheduling functions. It can also serve as a cyclic wake up from low-power modes without the need for external components.

9.1.2.26 Universal Serial Bus (USB) Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Universal Serial Bus (USB) module including USB flags and interrupts, seven USB endpoints, USB RAM and USB reset options. While all control transactions occur through bi-directional endpoint 0, the other endpoints can be set up for data transfer in the input or output direction.

Some of the microcontrollers in the HCS08 family contain USB compliant peripheral devices. These can be low-speed or high-speed USB slave devices. This means that all USB transfers are initiated by a host (i.e. a personal computer) and that the microcontroller needs to be set up to respond with the appropriate acknowledgement messages. According to the USB specification, there are a series of messages that go back and forth between the host and the device in order to set up and describe the channel for data transfer. All of these messages can be set up through the USB I/O commands built into the simulator. This section goes through an example of this, showing how the simulator can be used to test out code for driving the USB peripheral.

9.1.2.27 Voltage Reference Module

In Full Chip Simulation (FCS) mode, this module lets you simulate all the functionality of the Voltage Reference (VREF) module. The module is a bandgap buffer system intended to supply an accurate voltage output that is trimmable by an 8-bit register in 0.5 mV steps. It can be used internally for the analog peripherals of an ADC channel or for an ACMP input. VREF has three operating modes that provide different levels of load regulation and power consumption.

9.2 P&E Hardware Interface Connection for HCS08

This section describes the HCS08 P&E Connection options. The Connection setting permits a connection to HCS08 Freescale devices via P&E Multilink, Cyclone (including the Cyclone PRO, Cyclone Universal [FX]), and OSBDM hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

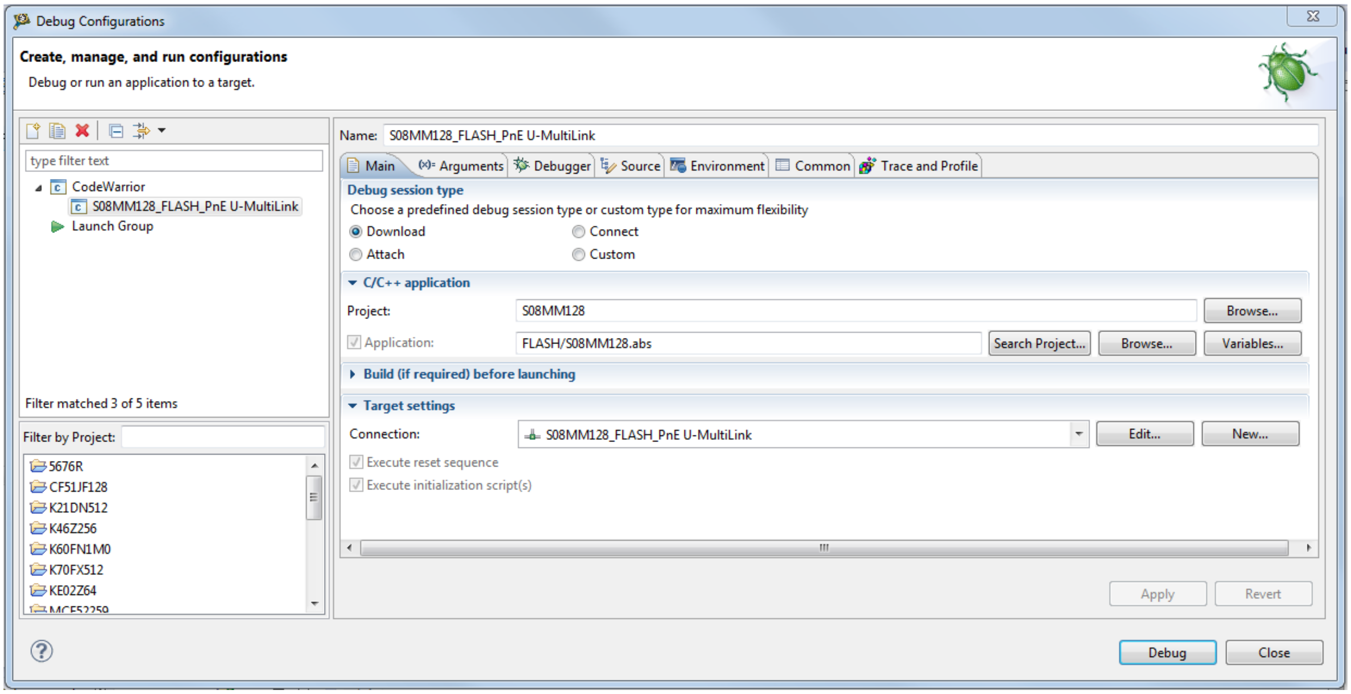
9.2.1 New Project Wizard

When creating a new project using the New Project Wizard,, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.

9.2.2 Launch Configuration Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project for which you would like to set the launch configurations.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - S08MM128_FLASH_PnE U-MultiLink
- Launch Group

Filter matched 3 of 5 items

Filter by Project:

- 5676R
- CF51JF128
- K21DN512
- K46Z256
- K60FN1M0
- K70FX512
- KE02Z64
- MCF52259

Name: S08MM128_FLASH_PnE U-MultiLink

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: S08MM128

☑ Application: FLASH/S08MM128.abs

Build (if required) before launching

Target settings

Connection: S08MM128_FLASH_PnE U-MultiLink

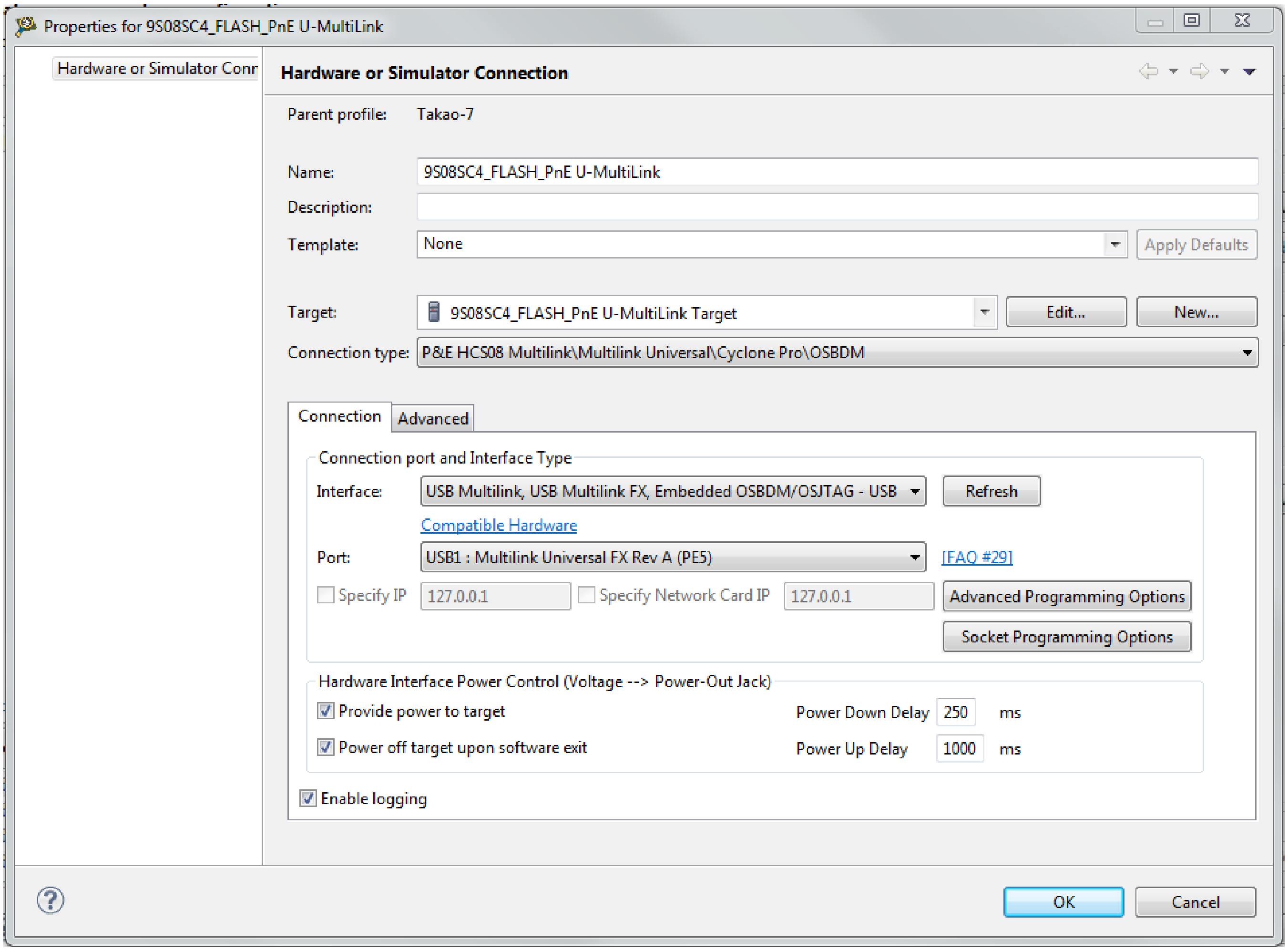
☑ Execute reset sequence

☑ Execute initialization script(s)

Figure 9-46. Debug Configuration Dialog Box

3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear.

5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Properties for 9S08SC4_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: 9S08SC4_FLASH_PnE U-MultiLink

Description:

Template: None

Target: 9S08SC4_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

[FAQ #29]

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

☑ Enable logging

Figure 9-47. Launch Configuration Dialog Box

9.2.3 Connection Options

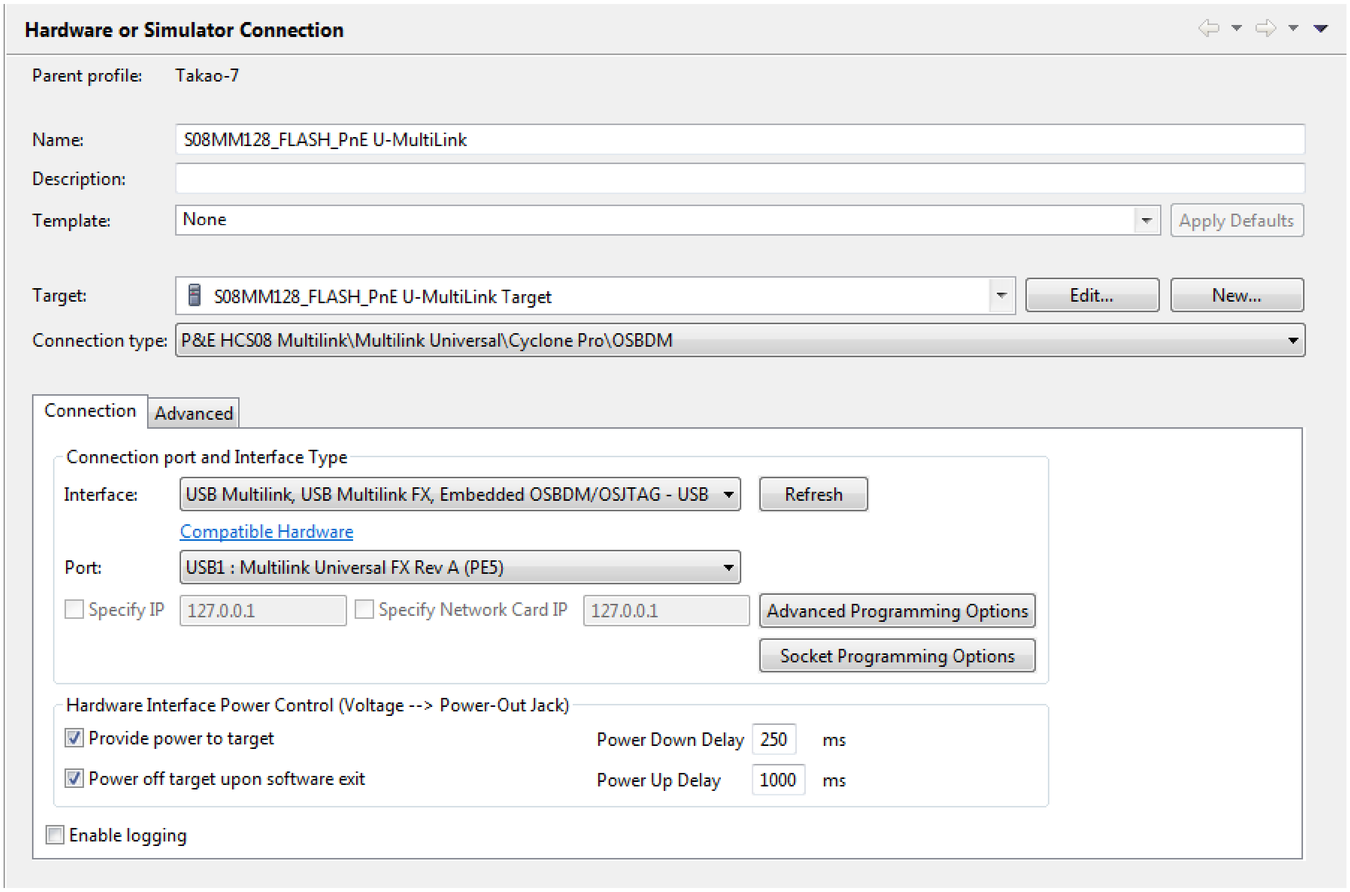
This topic describes all P&E HCS08 connection options, which are common to all P&E USB Multilink Universal [FX]/ USB Multilink, P&E Cyclone Serial, P&E Cyclone USB, P&E Cyclone Ethernet, and Open Source BDM connections.

The connection options include:

- Changing P&E Connections Settings
- Connection Assistant

9.2.3.1 Changing P&E Connections Settings

All connection settings for P&E hardware interfaces are configured using the **Connection** group in the **Main** tab of the **Debugger Configuration** dialog box.



Hardware or Simulator Connection

Parent profile: Takao-7

Name: S08MM128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S08MM128_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

□ Enable logging

Figure 9-48. P&E HCS08 Launch Configuration Dialog Box

The following table describes the options for the **P&E Multilink/Cyclone /OSBDM** connection.

Table 9-1. Connection Parameter Options for P&E Multilink/Cyclone/OSBDM

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB BDM Multilink (HCS08/HCS12/CFV1) - USB Port • USB Multilink Universal [FX] - USB Port NOTE: The USB Multilink Universal and the USB Multilink Universal [FX] can conveniently support all Freescale architectures found in the current CodeWarrior 10 version • Cyclone- Serial Port

Table continues on the next page...

Table 9-1. Connection Parameter Options for P&E Multilink/Cyclone/OSBDM (continued)

Option	Description
	• Cyclone- USB Port • Cyclone- Ethernet Port • OSBDM NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.
Socket Programming Options	The Socket Programming Options button brings up a dialog that provides you with a graphical representation of the signals that must be connected from the BDM header to the pins of the microprocessor, in order to use Freescale socket adapters.
Advanced Programming Options	The Advanced Programming Options button brings up a dialog that provides you with options to configure the flash programming procedure.
Specify IP (Cyclone Ethernet only)	Use this option to specify the IP address of a Cyclone outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port dropdown box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Provide power to target (Cyclone and USB Multilink Universal FX only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (Cyclone and USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Regulator Output Voltage (Cyclone and USB Multilink Universal FX only)	This option adjusts the output voltage that powers the hardware target. Select a voltage value from this option's list box.
Power down delay (Cyclone and USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (Cyclone and USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.

WARNING

An improper voltage setting can damage the board.

To change P&E Connections settings, perform these steps.

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

   NOTE

   It is assumed that you have created a project and built it.

2. Select **Run > Debug Configurations** from the main menu bar of the IDE.

   The **Debug Configurations** dialog box appears.

3. Expand the **CodeWarrior** tree control in the left pane and select the launch configuration you want to debug
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the **Target** is the correct microcontrollers you want to debug. Use the drop-down box or the **Edit** button to change this option.
7. In the Connection Type drop-down box, select P&E HCS08 Multilink/Cyclone/OSBDM. The P&E connections settings will appear below.
8. Click **Refresh** to scan valid interface and port.

   Valid interfaces and ports appear in the **Interface** and **Port** drop-down lists in the **Connection Port and Interface Type** group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

    NOTE

    The port displayed may vary depending on the interface. For example, if you select interface as Cyclone- Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Hardware Interface Power Control (Voltage --> Power -Out Jack) group.

NOTE

This group will be enabled for the Tracelink and USB Multilink Universal FX interfaces only. For USB Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

- Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
- Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.
- Select a voltage value from the Regulator Output Voltage drop-down list. This adjusts the output voltage that powers the hardware target.

WARNING

An improper voltage setting can damage the board.

- Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
- Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the hardware interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
- Click OK to save changes to the P&E Connections settings. The Connection Assistant dialog box will close.
- Click Close button to close the Debug Configuration dialog box.

9.2.3.1.1 P&E Hardware Interface Connection-Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]/ USB Multilink
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- Open Source BDM

9.2.3.1.1.1 P&E USB Multilink Universal [FX]/ USB Multilink

The P&E USB Multilink Universal [FX]/ USB Multilink Connection setting permits a connection to USB Multilink devices, which include the P&E BDM Multilink, USB Multilink Universal, and the USB Multilink Universal FX. P&E USB Multilink Universal [FX]/USB Multilink mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

9.2.3.1.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the **Main** tab, use the **Connection Selection** drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E HCS08 Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings..

To use P&E’s USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link to FAQ #29 to find popular solutions.

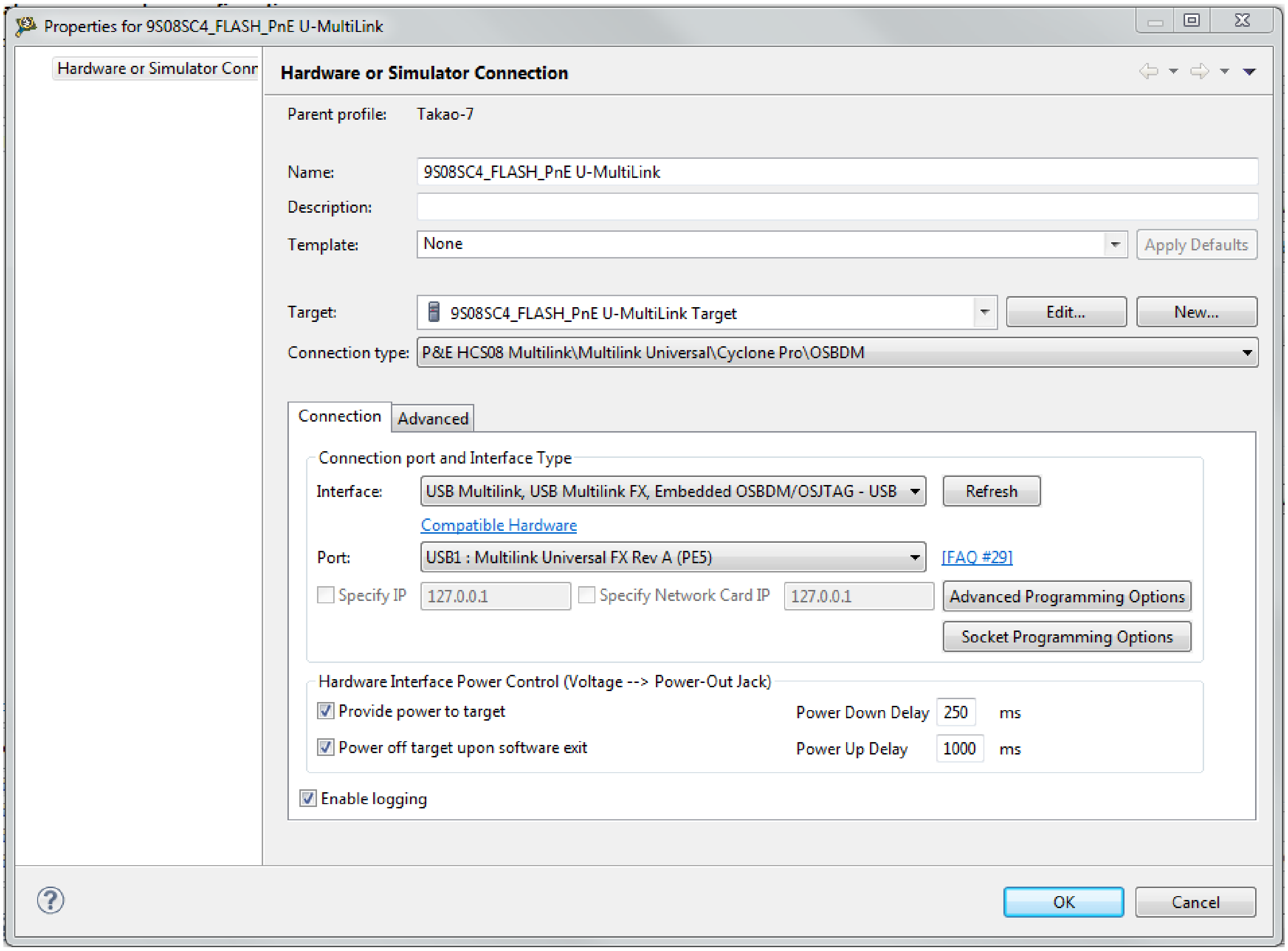
9.2.3.1.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

9.2.3.1.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the **Main** tab, use the **Connection Selection** drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E HCS08 Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for 9S08SC4_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: 9S08SC4_FLASH_PnE U-MultiLink

Description:

Template: None

Target: 9S08SC4_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

☑ Enable logging

Figure 9-49. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone – Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

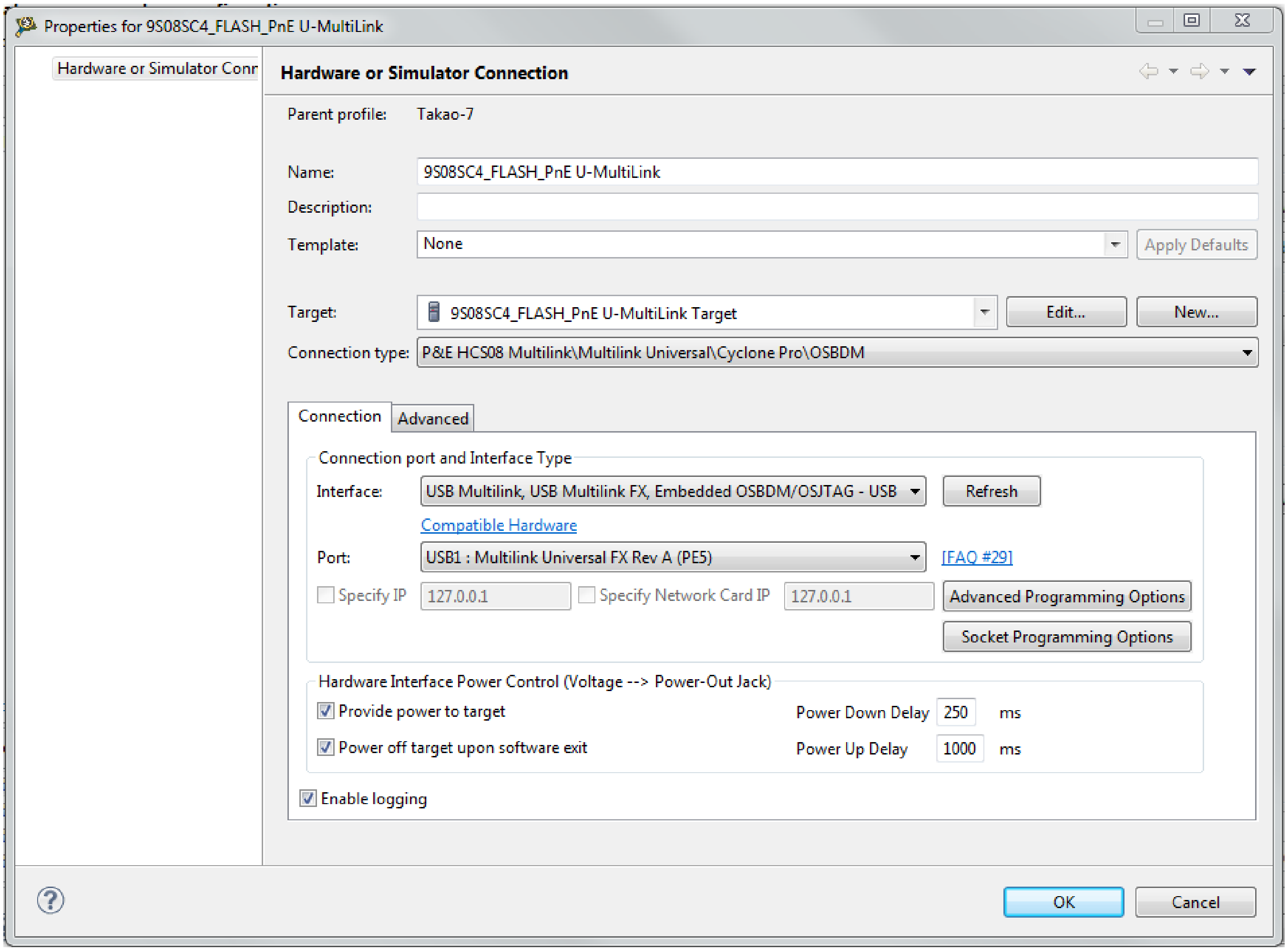
9.2.3.1.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources..

9.2.3.1.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the **Main** tab, use the **Connection Selection** drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E HCS08 Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for 9S08SC4_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: 9S08SC4_FLASH_PnE U-MultiLink

Description:

Template: None

Target: 9S08SC4_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

☑ Enable logging

Figure 9-50. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

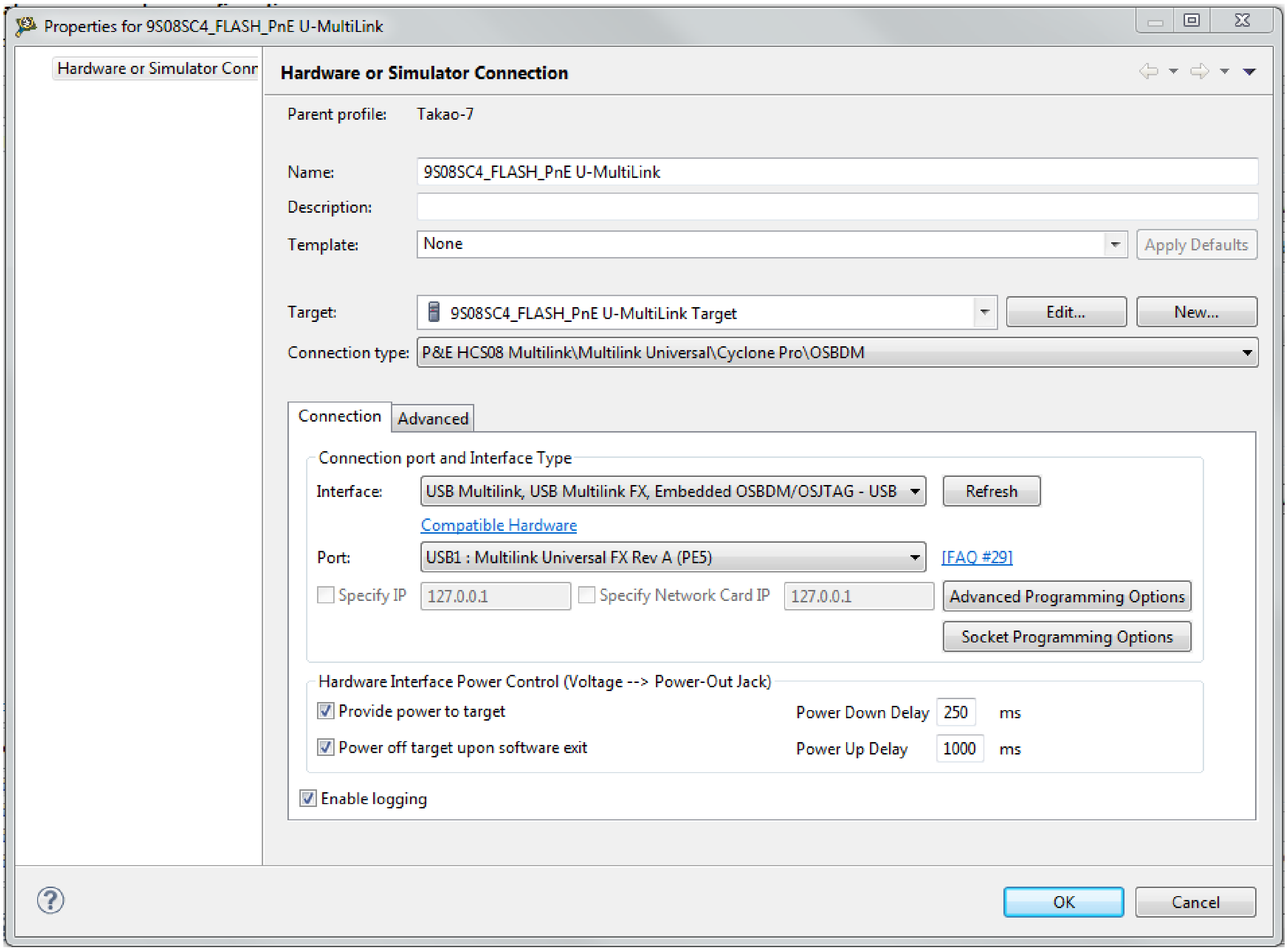
9.2.3.1.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

9.2.3.1.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the **Main** tab, use the **Connection Selection** drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E HCS08 Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for 9S08SC4_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: 9S08SC4_FLASH_PnE U-MultiLink

Description:

Template: None

Target: 9S08SC4_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

☑ Enable logging

Figure 9-51. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone – Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

9.2.3.1.1.5 Open Source BDM

Freescale supplies certain development boards with an integrated debug circuit based on Open Source BDM. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink. The development board also derives its power from the USB Bus.

The Open Source BDM circuit design (OSBDM-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

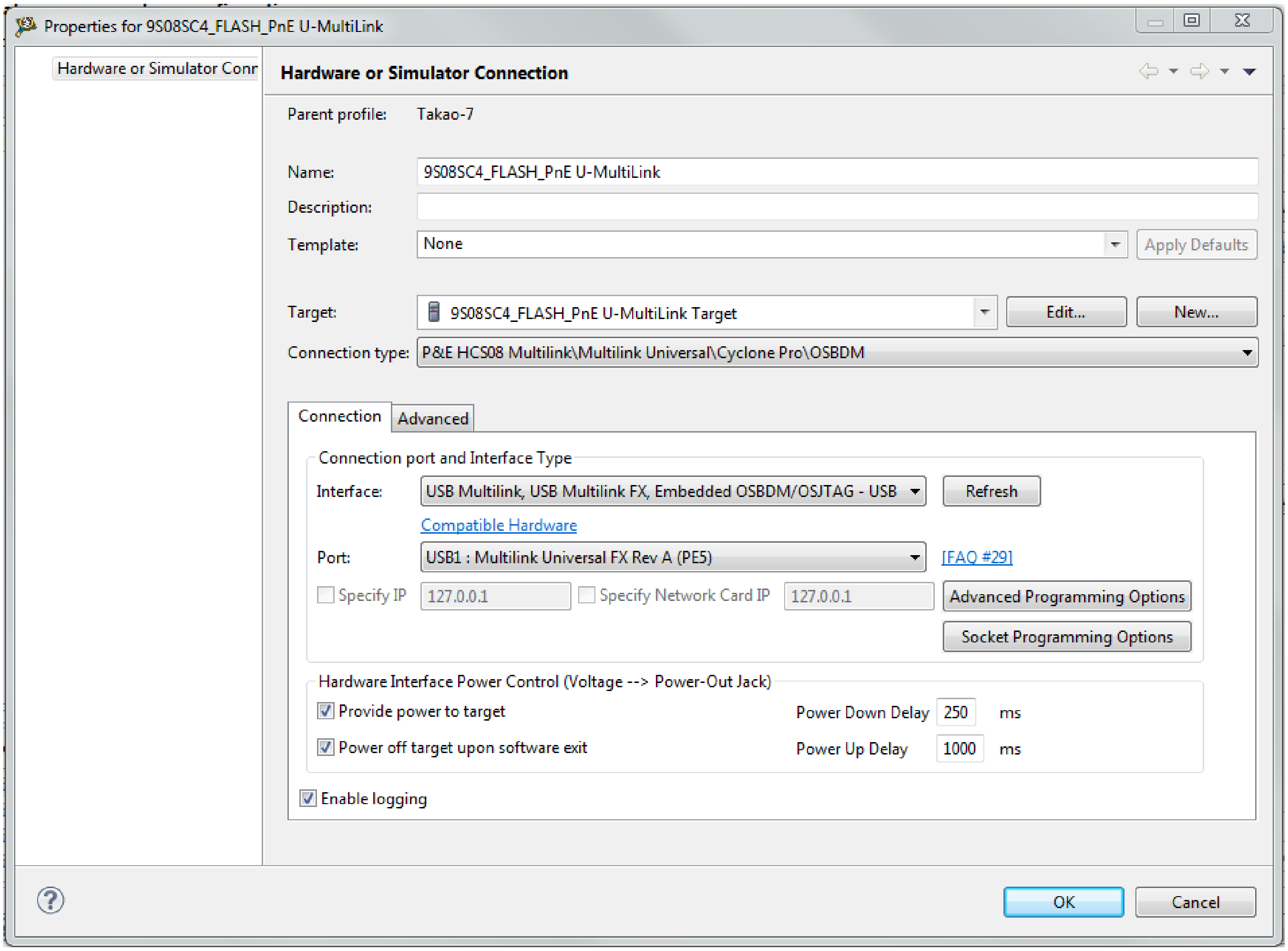
Integration with CodeWarrior is handled via the "Open Source BDM" connection. P&E has integrated the Open Source BDM support into the same connection that supports both the USB Multilink and the Cyclone. All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSBDM (albeit at a lower data rate).

The Open Source BDM Connection setting permits a connection to Open Source BDM devices. Open Source BDM mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

9.2.3.1.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the **Main** tab, use the **Connection Selection** drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E HCS08 Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.


Properties for 9S08SC4_FLASH_PnE U-MultiLink
Hardware or Simulator Conr
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
[FAQ #29]
Advanced Programming Options
Socket Programming Options
OK
Cancel

Hardware or Simulator Connection

Parent profile: Takao-7

Name: 9S08SC4_FLASH_PnE U-MultiLink

Description:

Template: None

Target: 9S08SC4_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

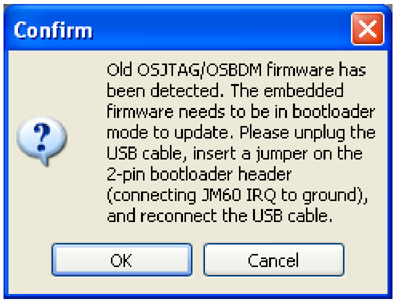
☑ Enable logging

Figure 9-52. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

9.2.3.1.1.5.2 OSBDM Firmware Update

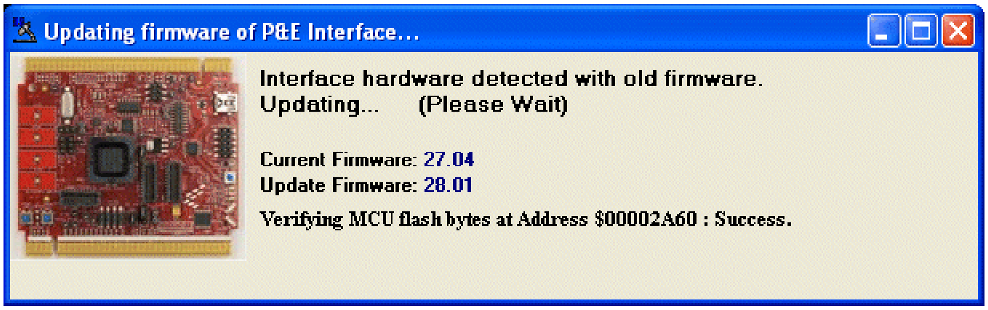
All CodeWarrior IDE's version 10.x and higher have an automatic firmware update mechanism for built-in OSBDM hardware interfaces. Whenever an OSBDM-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSBDM firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.



Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 9-53. Old OSBDM Firmware Detected

To update the firmware, the OSBDM device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSBDM-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSBDM device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSBDM device and click on the OK button. If done correctly, the automatic firmware update will occur.



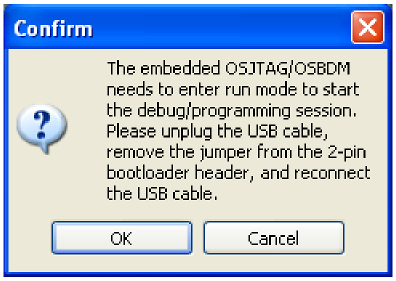
Interface hardware detected with old firmware.
Updating... (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01

Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 9-54. OSBDM Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSBDM device must exit Bootloader mode and enter into Run mode.



The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

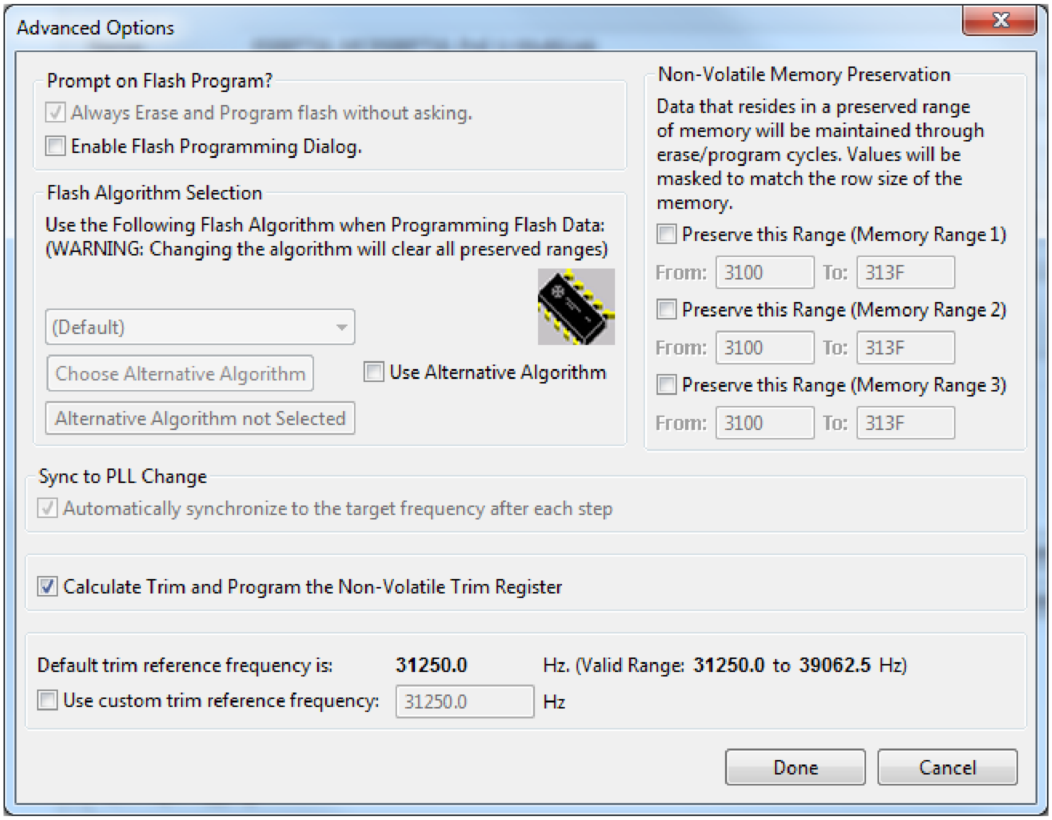
Figure 9-55. Start OSBDM Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSBDM device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSBDM firmware. If for any reason you experience difficulty performing OSBDM firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail-safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

9.2.3.1.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

□ Use Alternative Algorithm

Alternative Algorithm not Selected

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 3100 To: 313F

□ Preserve this Range (Memory Range 2)

From: 3100 To: 313F

□ Preserve this Range (Memory Range 3)

From: 3100 To: 313F

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

☑ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: **31250.0** Hz. (Valid Range: **31250.0** to **39062.5** Hz)

□ Use custom trim reference frequency: 31250.0 Hz

Figure 9-56. Advanced Options Dialog Box

9.2.3.1.2.1 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

9.2.3.1.2.1.1 Trim Options

The Calculate Trim and Program the Non-Volatile Trim Register checkbox enables automatic calculation and programming of the trim value to a designated Non-Volatile memory location.

9.2.3.1.2.1.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the entire EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase, and re-programmed immediately afterwards, thereby preserving the data in these ranges. Any attempt to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

9.2.3.1.2.1.3 Sync to PLL Change Checkbox

The debugger requires that Sync to PLL Change be selected to synchronize the software/hardware connection with the microprocessor during the Flash erasing/programming procedure. This option is always enabled for M68HCS08 devices.

9.2.3.1.2.1.4 Calculate and Program Non-Volatile Trim

The checkbox gives you the option of trimming device to default center frequency. If this checkbox is selected, a calculated trim frequency will be programmed to a dedicated non-volatile memory location during the next debugging session.

9.2.3.1.2.1.5 Custom Trim

When the checkbox is checked, you have the ability to input a custom center frequency within an allowed range for a given device. A trim value based on this frequency will be calculated and programming into dedicated non-volatile memory location during the next debug session.

9.2.3.1.2.1.6 Alternative Algorithm Functionality

Once you create a project for a specific HCS08/RS08/CFV1 microprocessor, the debugger specifies a default algorithm to use during all Flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the `<CW_Install>/MCU/bin/plugins/support/HC08/gdi/P&E directory`

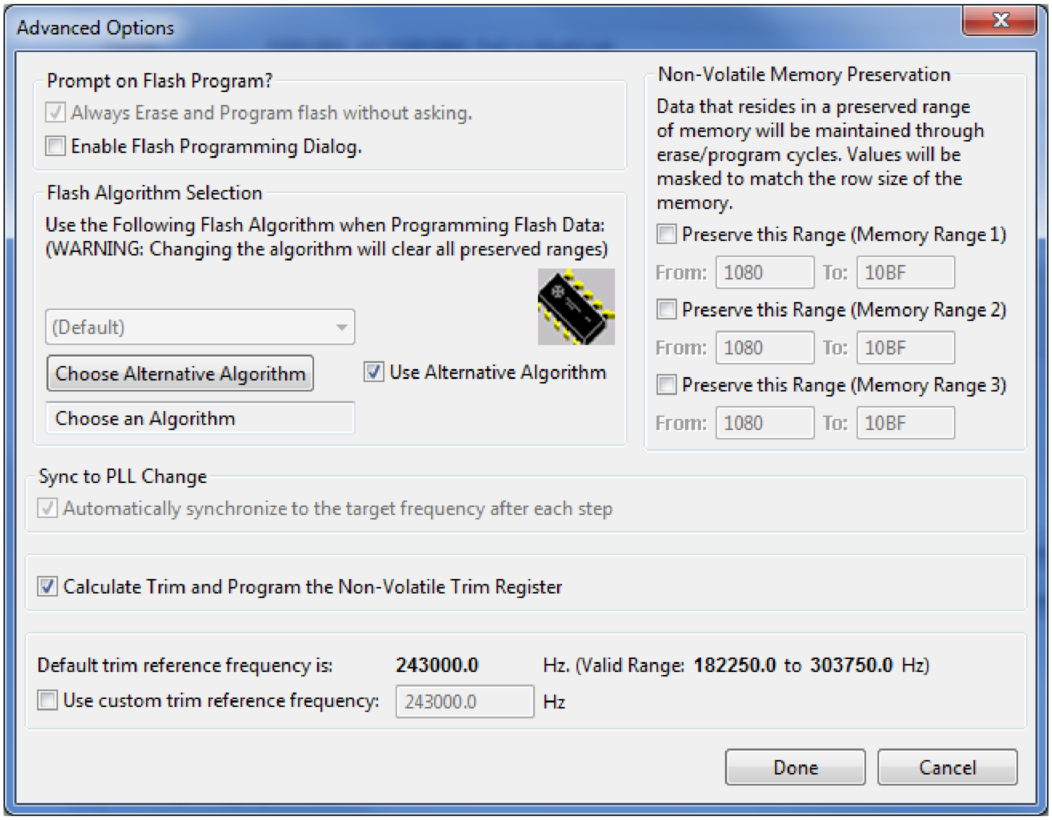
However, you can override the default algorithm via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. This feature can be used to select a custom programming algorithm, or select another one of P&E's many programming algorithms for use with a specific project.

Tip

Selecting a wrong programming algorithm may damage your device, lead to under/over programming situations, or simply not program portions of the project file. Therefore it is recommended to use the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. Check the **Use Alternative Algorithm** checkbox.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm ☑ Use Alternative Algorithm

Choose an Algorithm

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 1080 To: 10BF

□ Preserve this Range (Memory Range 2)

From: 1080 To: 10BF

□ Preserve this Range (Memory Range 3)

From: 1080 To: 10BF

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

☑ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: 243000.0 Hz. (Valid Range: 182250.0 to 303750.0 Hz)

□ Use custom trim reference frequency: 243000.0 Hz

Done Cancel

Figure 9-57. Advanced Options - Alternative Algorithm Checkbox

2. Click the **Choose Alternative Algorithm** button, which lets you browse for an alternative algorithm.

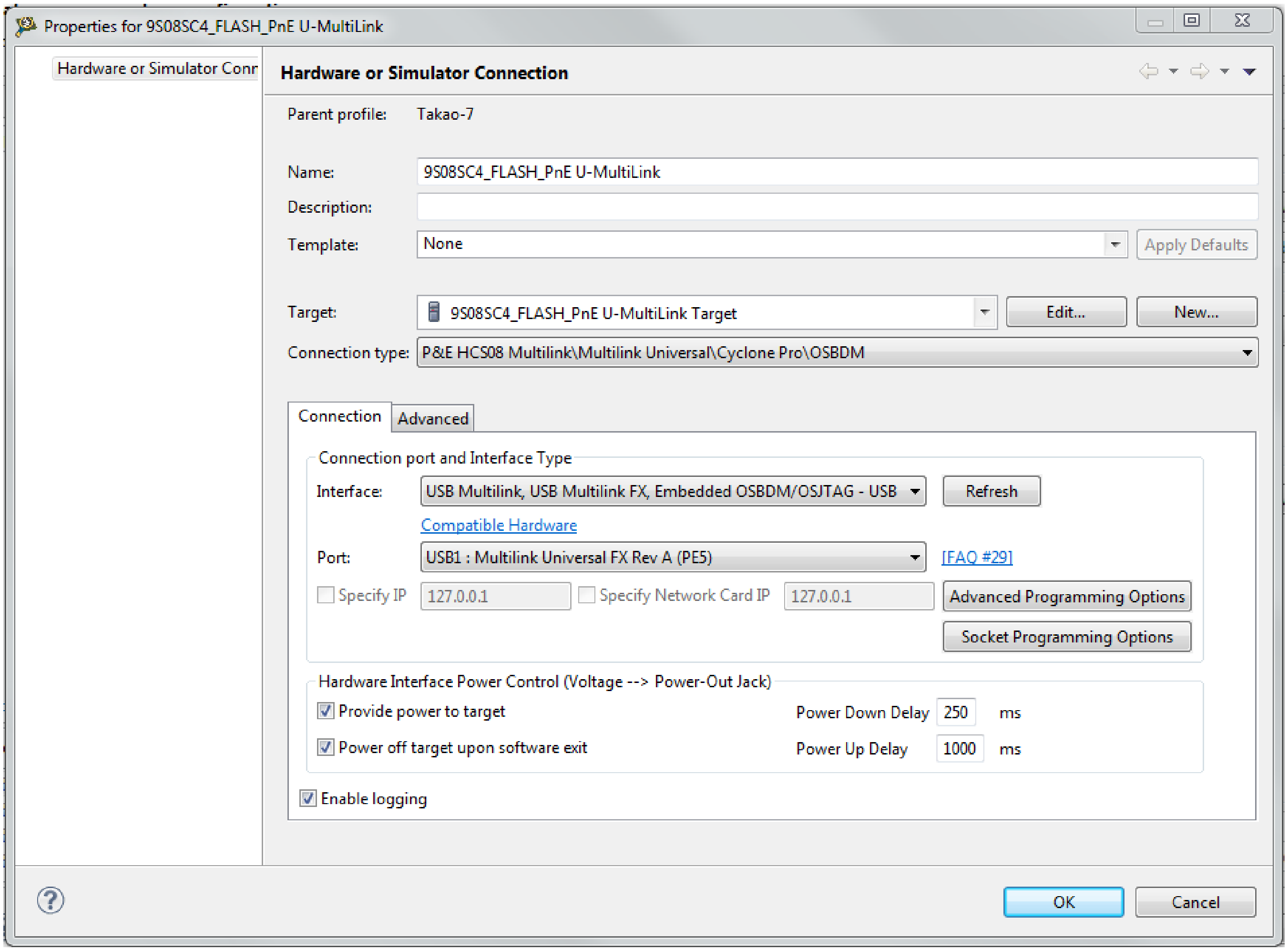
3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the Choose Alternative Algorithm button.

At this point, the current project performs all future Flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

9.2.3.1.3 Socket Programming Options Button

The Programming Adapter Connections dialog assistant is designed to facilitate the use of an extensive set of Freescale programming socket adapters. This dialog can be used to get a graphical representation of the signals that must be connected from the BDM header to the pins of the microprocessor. Making these connections lets you establish communication with a given device via a hardware debug interface.

The Socket Programming Options button in the BDM Launch Configuration dialog box takes you to the Programming Adapter Connections dialog box, where you can look up pin connection settings for the selected package type of the target processor. Only available package types for each target processor are listed in the Package drop-down list. Once you have selected a package type, the Adapter Information section provides the part number of the adapter board, the socket number where the processor should be placed, and a pair of header numbers that indicate which connections should be made between them. Immediately below the Adapter Information section you will find a pin layout that displays the required connections between the aforementioned pair of headers.



Properties for 9S08SC4_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: 9S08SC4_FLASH_PnE U-MultiLink

Description:

Template: None

Target: 9S08SC4_FLASH_PnE U-MultiLink Target

Connection type: P&E HCS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

[FAQ #29]

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

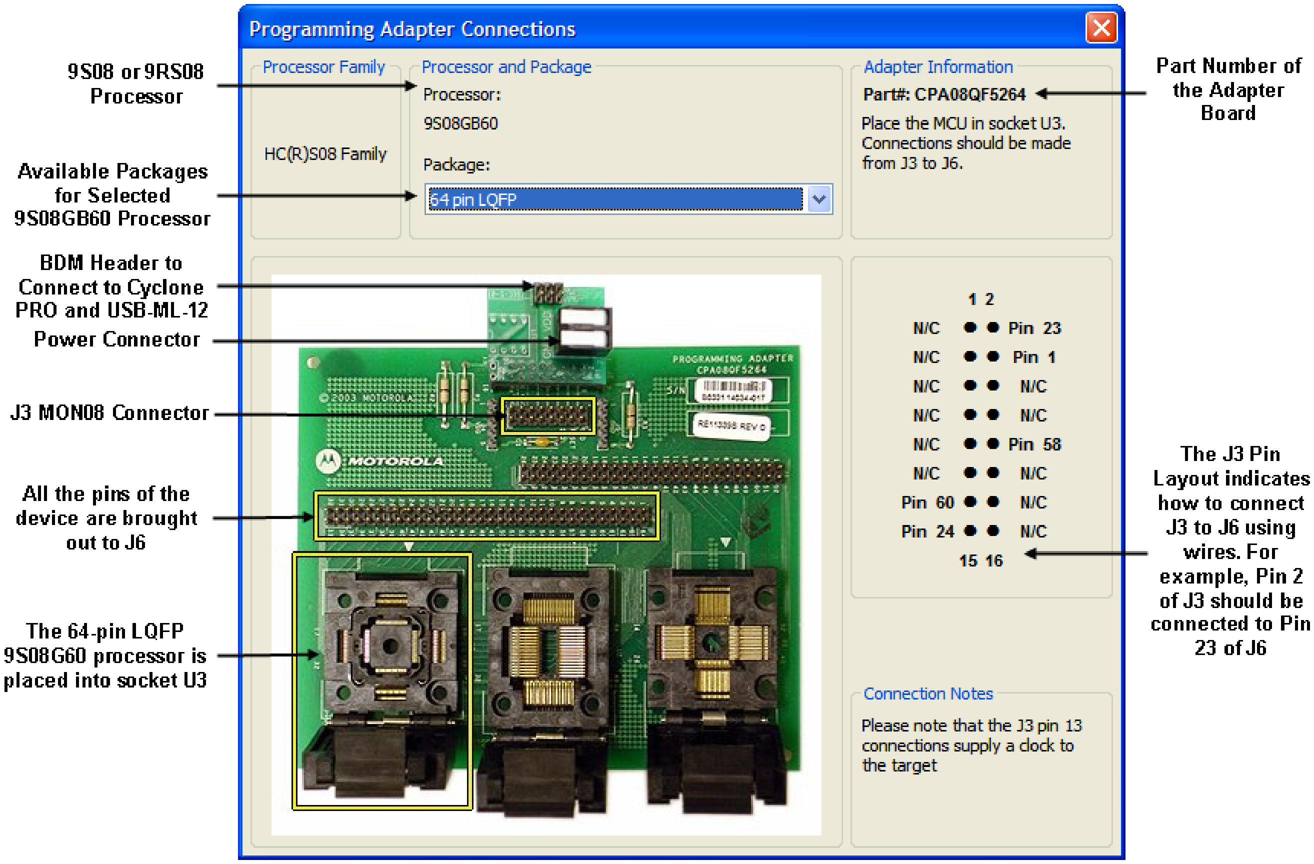
Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

☑ Enable logging

Figure 9-58. HCS08 BDM Launch Configuration Dialog Box



9S08 or 9RS08 Processor

Available Packages for Selected 9S08GB60 Processor

BDM Header to Connect to Cyclone PRO and USB-ML-12

Power Connector

J3 MON08 Connector

All the pins of the device are brought out to J6

The 64-pin LQFP 9S08G60 processor is placed into socket U3

Part Number of the Adapter Board

The J3 Pin Layout indicates how to connect J3 to J6 using wires. For example, Pin 2 of J3 should be connected to Pin 23 of J6

Programming Adapter Connections

Processor Family

HC(R)S08 Family

Processor and Package

Processor:

9S08GB60

Package:

64 pin LQFP

Adapter Information

Part#: CPA08QF5264

Place the MCU in socket U3. Connections should be made from J3 to J6.

	1	2	
N/C	●	●	Pin 23
N/C	●	●	Pin 1
N/C	●	●	N/C
N/C	●	●	N/C
N/C	●	●	Pin 58
N/C	●	●	N/C
Pin 60	●	●	N/C
Pin 24	●	●	N/C
	15	16	

Connection Notes

Please note that the J3 pin 13 connections supply a clock to the target

Figure 9-59. Programming Adapter Connections Dialog Box

9.2.3.2 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug and the program cannot connect to the interface hardware specified in the Launch Configuration dialog box. To edit or change your debugger connection:

1. Select the P&E device that you are using from the first drop-down list and click Refresh..
2. Using the second drop-down list, select the port on which the interface is connected.
3. Use the Cyclone Power Control panel to configure the power and delay settings (Cyclone only).
4. Click the Retry button.

9.2.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.

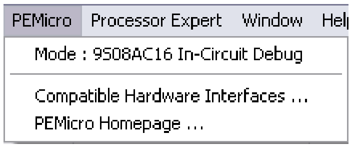

PEMicro
Processor Expert
Window
Help
Mode : 9S08AC16 In-Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 9-60. Additional Connection Menu Options

Chapter 10
Connections - RS08

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the RS08 full chip simulator or the target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with simulation platforms and hardware devices that are part of the RS08 device family.

The topics in this chapter are:

- P&E Full Chip Simulation
- P&E Hardware Interface Connection for RS08

10.1 P&E Full Chip Simulation

This topic describes the settings of the connections that interface the CodeWarrior debugger with the RS08 simulator.

10.1.1 Create New Connection for Full Chip Simulation

Full Chip Simulation (FCS) connection runs a complete simulation of all processor peripherals and I/O on your personal computer. Thus, when debugging an FCS project for a selected derivative, it is not necessary to connect your PC with a Microcontrollers development or target board.

To change connection in the IDE, perform these steps.

1. Right-click on your Project > Properties.

   The Properties windows appears.

2. Select Run/Debug Settings and click on the New... button.
3. Select CodeWarrior Download and click OK.

   The Edit Configuration window appears.

4. By default the project and application is already set. Change the name of your connection if you wish. You will need to create a new Connection. Within the Connection section, click on the New... button.

   The New Connection Wizard will appear.

5. Open the CodeWarrior Bareboard Debugging and select Hardware or Simulator Connection. Click Next.
6. Give your new connection a name. For connection type, change the setting to P&E RS08 FCS for Full Chip Simulation.
7. You will need to select the Target device. You can either select a pre-existing target or create a New target. When complete, click on the Finish button.
8. The wizard creates a simulator project for the HCS08 architecture according to your specifications. You can access and edit the project connections by right-click on your project > Debug As > Debug Configurations.

10.1.2 Module Options

The PEMicro menu includes the Full Chip Simulation options for the modules that have specialty commands associated with them for a chosen device.

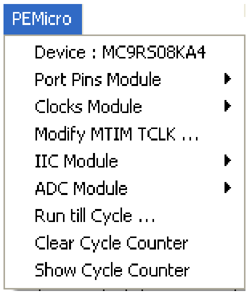


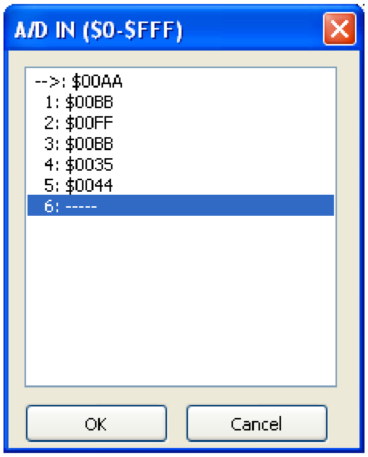
Figure 10-1. PEMicro Menu

The options available are:

- ADC Module
- Internal Clock Source Module
- Inter-Integrated Circuit Module Option
- Keyboard Interrupt Module
- Liquid Crystal Display Driver Module Option
- Modulo Timer Interrupt Module
- Input/Output (I/O) Ports Module
- Serial Communications Interface Module
- Serial Peripheral Interface Module
- Timer Interface Module

10.1.2.1 ADC Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Analog to Digital Conversion (ADC) module including data input on all ADC channels, flag polling, interrupt operation, and the bus and CGMXCLK reference clock sources. FCS mode uses the buffered input structure to simulate the ADC inputs. You can queue up to 256 data values. To queue the ADC Input Data, use the ADDI command in the command prompt. If the data parameter is given, the value is placed into the next slot in the input buffer. Otherwise, if no parameter is provided, a window is displayed with the input buffer values. Input values can be entered while the window is open. An arrow points to the next value to be used as input to the ADC. The conversion takes place after a proper value is written to the ADC Status and Control register. Once the conversion occurs, the arrow moves to the next value in the ADC Buffer.

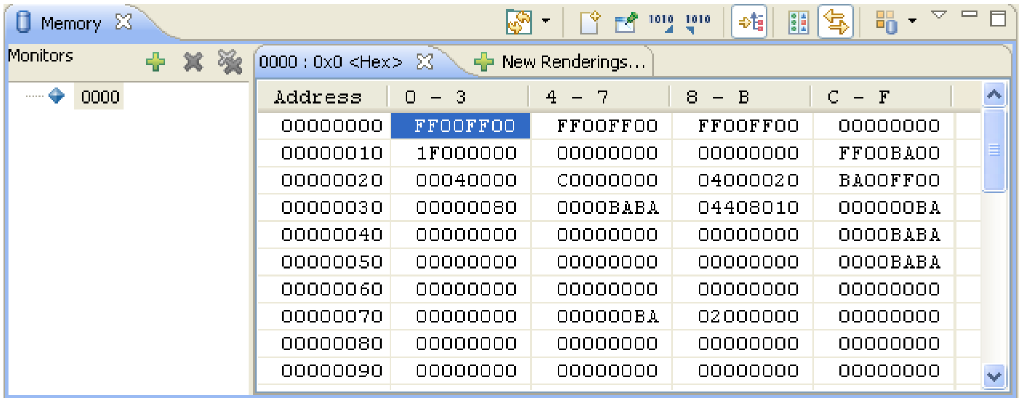


```
-->: $00AA
 1: $00BB
 2: $00FF
 3: $00BB
 4: $0035
 5: $0044
 6: -----
```

Figure 10-2. ADC IN Buffer Display

The ADCLR command can be used at any point to flush the input buffer for the ADC simulation.

After the conversion is complete, the first queued value is passed from the data buffer into the ADC data register. It can be observed in the Memory window by displaying the memory location corresponding to the ADC data register.



```
Address    0 - 3     4 - 7     8 - B     C - F
00000000   FF00FF00  FF00FF00  FF00FF00  00000000
00000010   1F000000  00000000  00000000  FF00BA00
00000020   00040000  C0000000  04000020  BA00FF00
00000030   00000080  0000BABA  04408010  000000BA
00000040   00000000  00000000  00000000  0000BABA
00000050   00000000  00000000  00000000  0000BABA
00000060   00000000  00000000  00000000  00000000
00000070   00000000  000000BA  02000000  00000000
00000080   00000000  00000000  00000000  00000000
00000090   00000000  00000000  00000000  00000000
```

Figure 10-3. Memory Component Window

When the conversion is complete, FCS sets the appropriate flag. If interrupts are enabled, the Program Counter changes flow to the interrupt routine (as defined in the vector space of the MCU).

NOTE

For more information on ADC configuration, refer to the Freescale user manual for your microprocessor.

10.1.2.1.1 ADC Module Commands

The following commands are available for the RS08/HCS08 ADC Module.

10.1.2.1.1.1 ADDI Command

The ADDI command lets you input the data into the ADC converter. If a data parameter is given, the value is placed into the next slot in the input buffer. Otherwise, if no parameter is given, a window is displayed with the input buffer values. Input values can be entered while the window is open. An arrow points to the next value to be used by the ADC. The maximum number of input values is 256 bytes.

Syntax

```
>gdi ADDI [<n>]
```

Where: `<n>` The value to be entered into the next location in the input buffer.

Example

```
>gdi ADDI $55
```

Set the next input value to the ADDI to $55

```
>gdi ADDI
```

Pull up the data window with all the input values.

10.1.2.1.1.2 ADCLR Command

Use the ADCLR command to flush the input buffer for ADC simulation. This resets the input data buffer and clears out all values. Notice that if the ADC is currently using a value, this command does not prevent the ADC from using it. Refer to ADDI command for information on how to access the input buffer of the ADC interface.

Syntax

```
>gdi ADCLR
```

Example

```
>gdi ADCLR
```

Clear the input buffer for ADC simulation.

10.1.2.2 Internal Clock Source Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Internal Clock Source (ICS) Module, including:

- Phase Locked Loop (PLL) generation
- Automatic lock detection
- Interrupt
- Acquisition
- Tracking
- Flag polling

FCS mode uses a simulated External Oscillator Frequency change command (XTAL) lets you input the desired XTAL value. To check the current value of the External Oscillator, Bus Frequency and ICSCLK Frequency, open the RS08FCS menu and select Clocks Module > Show MCU Clocks.

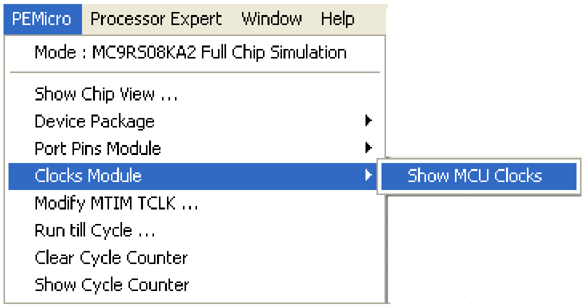
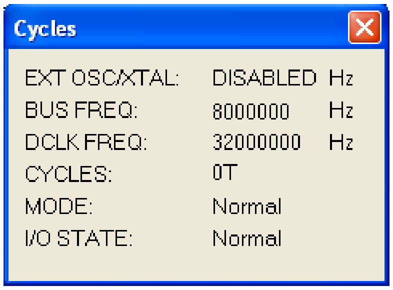


Figure 10-4. Clocks Module Extended Menu

Once you select the MCU Clocks menu, the Cycles window displays all of the aforementioned Clock Frequencies, or you can select the Show Cycle Counter option within the FCS menu to get the same window.



EXT OSC/XTAL:	DISABLED	Hz
BUS FREQ:	8000000	Hz
DCLK FREQ:	32000000	Hz
CYCLES:	0T	
MODE:	Normal	
I/O STATE:	Normal	

Figure 10-5. Frequency Display

Within the FCS menu, you can select the Run till Cycle option, which lets you begin code execution and stop execution when the specified cycle count is reached. Note that the parameter given is not the number of cycles that executed, but rather the total cycle-count of the simulator (displayed in the Register Window).

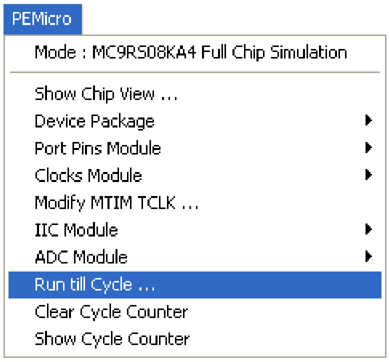
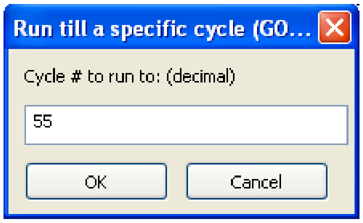


Figure 10-6. Run till Cycle command

This command is extremely useful for verifying specific timings of a given event, running until a given event is complete, or just before it completes to enable stepping through the event or any application where cycle-timed execution is desired.

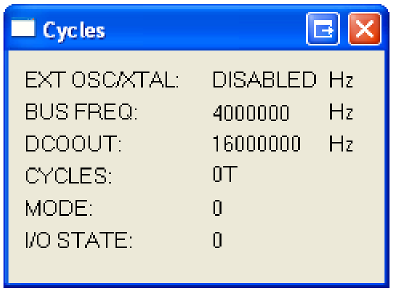

Run till a specific cycle (GO...

Cycle # to run to: (decimal)

55

Figure 10-7. Run till Cycle Dialog Box

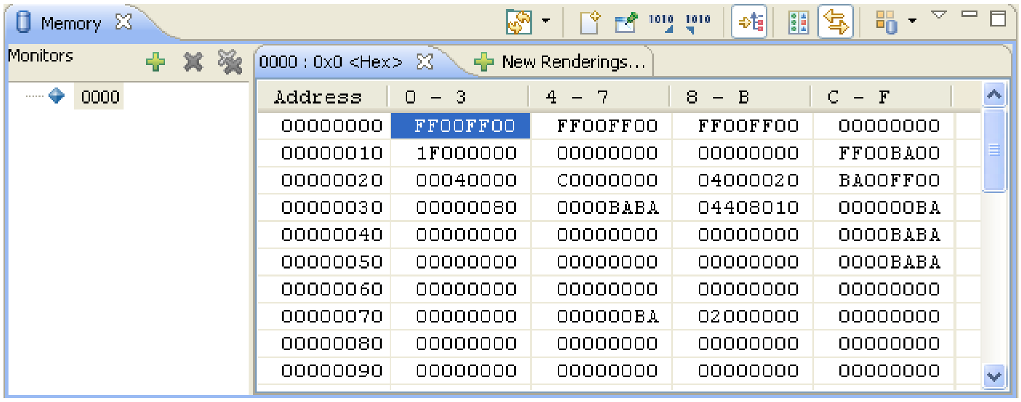
You can also select the Clear Cycle Counter option within the FCS menu, which clears the cycle counter. If you select the Show Cycle Counter option within the FCS menu, you can check to make sure that the cycle counter is zero.


Cycles

EXT OSC/XTAL:	DISABLED	Hz
BUS FREQ:	4000000	Hz
DCOOUT:	16000000	Hz
CYCLES:	0T	
MODE:	0	
I/O STATE:	0	

Figure 10-8. Cycle Counter Dialog Box with Cleared Counter

Once the ICG is properly configured, you can monitor the status of the PLL by polling the corresponding flag. If PLL interrupt is enabled, FCS jumps to an appropriate subroutine, as long as the interrupt vector is properly defined. To observe the flag going up as a result of the corresponding CPU event, situate your Memory window on the memory location of the ICG Status and Control register.



Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 10-9. Memory Window

For more information on how to properly configure Clock Generation, refer to the Freescale reference manual for your microprocessor.

10.1.2.2.1 Internal Clock Source Commands

The following commands are available for the RS08 Internal Clock Source Module.

10.1.2.2.2 XTAL Command

Use the XTAL command to change the value of the simulated external oscillator. This in turn affects the input to the PLL/DCO, and therefore the bus frequency. The P&E simulator is a cycle-based simulator, so changing the XTAL value does not affect the speed of simulation. It does, however, affect the ratio in which peripherals receive cycles. Certain peripherals that run directly from the XTAL will run at different speeds than those that run from the bus clock.

Syntax

```
>gdi XTAL <n>
```

Where: ï¿½ `<n>`, by default, is a hexadecimal number, representing the simulated frequency of an external oscillator. Adding the suffix `` `t' `` to the `'n'` parameter forces the input value to be interpreted as base 10.

Example

```
>gdi XTAL
```

Brings up an input window. The default base for this input value is 10. However, this value can be forced to a hexadecimal format through use of the suffix `'h'`.

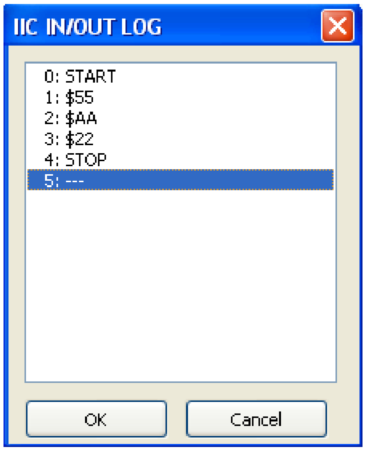
10.1.2.3 Inter-Integrated Circuit Module Option

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Inter-Integrated Circuit (IIC) module including:

- Flag polling
- Interrupt enabled mode
- Transmission and reception of external data
- Master and slave modes of operation
- START and STOP signal generation detection
- Acknowledge bit generation detection

FCS mode uses the buffered input/output structure to simulate IIC inputs. You can queue up to 256 data bytes into the input buffer. The output buffer of the USB module can also hold 256 output bytes. To queue the IIC Input Packets, use the IICDI <...> command in the command prompt. For a more detailed description of the command, refer to the IIC Commands section. If the IIC packet parameters are properly defined, the packet is placed into the next slot in the input buffer. Otherwise, if no parameters are provided, an IIC Input Buffer window is displayed.

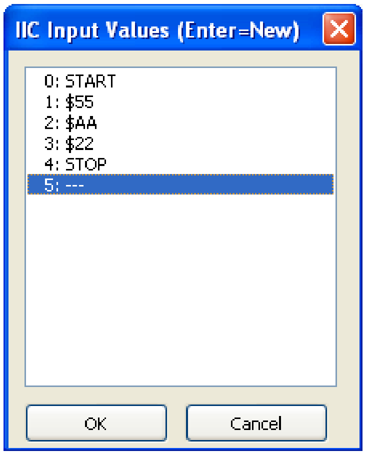
You can enter different IIC packet parameters while the window is open, including START, STOP, ACK, NACK and data bytes. An arrow points to the next byte to be used as input to the IIC. The data from the IIC input buffer is written to the IIC module registers once the IIC module is turned on and properly configured for receiving data from an external IIC device. Once simulation of the data transmission is over, the arrow moves to the next value in the IIC Input Buffer.



```
0: START
1: $55
2: $AA
3: $22
4: STOP
5: ---
```

Figure 10-10. IIC Input Buffer Display

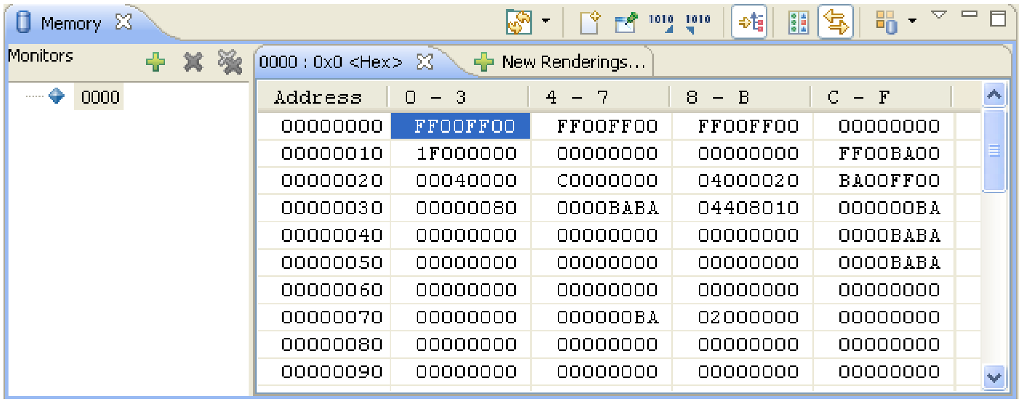
The IIC data input/output log buffer simulation lets you gain access to the past 256 IIC data bytes that have been shifted in and out of the module. To bring up the IIC IN/OUT LOG buffer dialog box, use the IICDO command.



0: START
1: $55
2: $AA
3: $22
4: STOP
5: ---

Figure 10-11. IIC IN/OUT LOG Buffer Display

The IICCLR command may be used at any point to flush the input as well as input/output log IIC buffers. After the IIC simulated input is received, the first queued-in data byte is passed from the data buffer into the corresponding IIC module registers. It can be observed in the Memory window by displaying the appropriate register location there.



Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 10-12. Memory Component Window

You can also observe different IIC flags in the Memory window. If you run the module in Flag Polling mode, poll the flag corresponding to the expected IIC event. If the IIC interrupts are enabled, FCS jumps to an appropriate subroutine as long as the IIC interrupt vectors are properly defined.

NOTE

For more information on how to configure IIC module for desired operation, refer to the Freescale user manual for your microprocessor.

10.1.2.3.1 Inter-Integrated Circuit Module Commands

The following commands are available for the RS08 Inter-Integrated Circuit (IIC) module. Command function is identical even though the module names differ.

10.1.2.3.2 IICDI Command

The IICDI command lets you input data into a buffer of data to shift into the IIC module when it receives data from an external device. If a data parameter is given, the value is placed into the next slot in the input buffer. Otherwise, if no parameter is given, a window is displayed with the input buffer values. Input values can be entered while the window is open. The maximum number of input values is 256. This command is useful for either inputting response data from a slave target or for inputting data packets from an external master. Note that when the microprocessor attempts to read an acknowledge from an external device, and the next value in the buffer is neither ACK nor NACK, the microprocessor automatically receives an ACK signal (i.e. assumes ACK unless NACK is specified).

Syntax

```
>gdi IICDI [<n>][START][STOP][ACK][NACK]
```

Where: ï¿½ `<n>` indicates the value to be entered into the next location in the input buffer

ï¿½ `START` indicates the incoming START signal

ï¿½ `STOP` indicates the incoming STOP signal

ï¿½ `ACK` corresponds to ACK signal

ï¿½ `NACK` corresponds to NACK signal

NOTE

> For a detailed description of the IIC protocol and a proper way to configure the IIC module, refer to the Freescale user manual for your microprocessor.

Eaxmple

```
>gdi IICDI
```

Pulls up the data window with all the input values

```
>gdi IICDI 22 33
```

This is an example of data being returned from a slave device. Once the MCU transmits a start signal and the target address, it receives an ACK from the slave device. An ACK is implied unless a NACK is specified via the IICDI command. The next two data bytes read are 22 and 23. If the microprocessor attempts to read another byte, it gets an $FF value followed by a NACK signal (NACK because nothing remains in the input buffer). The receiving device then generates a STOP signal. A more exact input from a device designed to return two bytes is:

```
>gdi IICDI ACK 22 ACK 23 NACK
```

IIC in master mode transmits to a slave:

ï¿½ If the slave device acknowledges all output bytes of the transmitting device, there is no need to specify an input packet. If the master device is going to transmit an address and two bytes, the following packet is equivalent to no packet:

```
>gdi IICDI ACK ACK ACK
```

ï¿½ If, however, the slave receiver is designed to generate a NACK signal after the second received data byte, the proper response packet is:

```
>gdi IICDI ACK ACK NACK
```

ï¿½ The address result being the first ACK, the first data result being the second ACK, and the second data byte being the NACK.

IIC in MASTER mode is not acknowledged by any Slave:

```
>gdi IICDI NACK
```

ï¿½ If the NACK signal is entered before the master device transmits a START signal, then the master device gets a NACK when it tries to read an acknowledge after the address is output. The master device then generates a STOP signal and releases the BUS.

IIC in SLAVE mode receives a Write from an external Master:

This example is for an external master that is writing to the microprocessor configured to simulate the slave mode operation. The packet contains both START and STOP signals which puts the simulated device into the slave mode.

```
>gdi IICDI START 55 AA 22 STOP
```

This input adds five values to the input queue, which is a packet from an external master, including the following procedure values:

ï¿½ A start signal comes in

ï¿½ The address $55 comes in, specifying a write (slave receive). The Address Register in the current simulated device has been previously set to $55

ï¿½ The data byte $AA comes in

ï¿½ The data byte $22 comes in

ï¿½ A STOP signal comes in

10.1.2.3.3 IICDO Command

The IICDO command displays a window, which shows data shifted in as well as shifted out of the IIC peripheral. An arrow points to the last output value transmitted/received. The maximum number of output values that the buffer can hold is 256.

Syntax

```
>gdi IICDO
```

Example

```
>gdi IICDO
```

View data from the input/output log buffer for IIC simulation.

10.1.2.3.4 IICCLR Command

Use the IICCLR command to flush the input and output buffers for IIC simulation. This resets the buffers and clears all values. Notice that if the IIC is currently shifting a value, this command does not prevent the IIC from finishing the transfer.

Syntax

```
>gdi IICCLR
```

Example

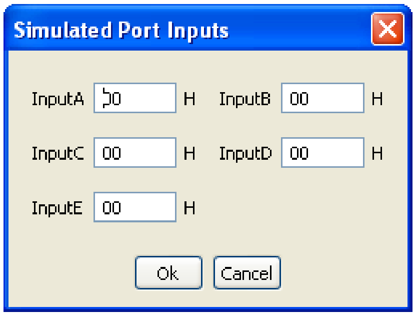
```
>gdi IICCLR
```

Clear input and output buffers for IIC simulation.

10.1.2.4 Keyboard Interrupt Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Keyboard Interrupt (KBI) module, including the edge-only, edge and level interrupt, and flag polling modes of operation. FCS mode uses simulated port inputs to trigger the KBI event from the proper I/O port pin.

To define an input state of the specific port, enter the INPUT<x> <n> command in the Command window. The <x> represents the corresponding I/O port, while <n> stands for the input value to write to this port. At the same time, you can use the INPUTS command to bring up the Simulated Port Inputs for all general I/O ports. It displays the current simulated values to all applicable input ports. Refer the documentation for Timer Module Commands for more information about the various forms of this command.



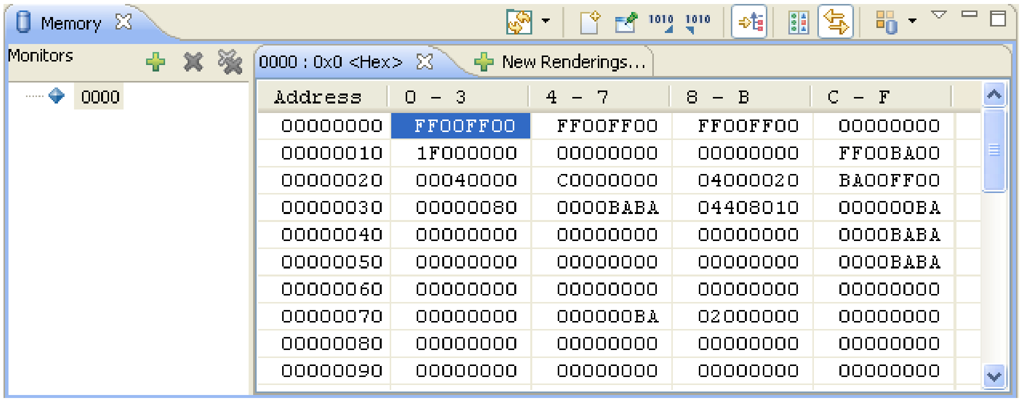
InputA 00 H InputB 00 H

InputC 00 H InputD 00 H

InputE 00 H

Figure 10-13. Simulated Port Inputs Dialog Box

Use the Simulated Port Inputs dialog box to reconfigure the input value to any I/O port. To trigger the event, manipulate the inputs to the port in the appropriate manner, depending on whether the KBI is configured for edge-only or edge and level. Once the KBI event takes place, you can observe the KEYF Flag bit, which is a part of the Keyboard Status and Control register, in the Memory window.



```
Address    0 - 3      4 - 7      8 - B      C - F
00000000   FF00FF00   FF00FF00   FF00FF00   00000000
00000010   1F000000   00000000   00000000   FF00BA00
00000020   00040000   C0000000   04000020   BA00FF00
00000030   00000080   0000BABA   04408010   000000BA
00000040   00000000   00000000   00000000   0000BABA
00000050   00000000   00000000   00000000   0000BABA
00000060   00000000   00000000   00000000   00000000
00000070   00000000   000000BA   02000000   00000000
00000080   00000000   00000000   00000000   00000000
00000090   00000000   00000000   00000000   00000000
```

Figure 10-14. Memory Component Window

You can poll the KBI Interrupt Pending flag if the Polling Mode is simulated. In Interrupt Mode, the simulator branches to an appropriate interrupt subroutine as long as the KBI interrupt vector is properly configured.

NOTE

For more information on KBI configuration, refer to the Freescale user manual for your microprocessor.

10.1.2.4.1 Keyboard Interrupt Commands

Use the following commands for Keyboard interrupt manipulation.

10.1.2.4.2 INPUT<x> Command

The INPUT<x> command sets the simulated inputs to port <x>. The CPU reads this input value when port <x> is set as an input port.

Syntax

```
>gdi INPUT<x> <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` is an eight-bit simulated value for port <x>

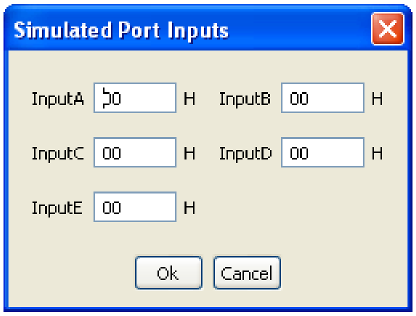
Example

```
>gdi INPUTA AA
```

Simulate the input AA on port A.

10.1.2.4.3 INPUTS Command

In FCS and CPU-Only Simulation mode, the INPUTS command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



Simulated Port Inputs

InputA 00 H InputB 00 H

InputC 00 H InputD 00 H

InputE 00 H

Figure 10-15. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values to any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

Show I/O port input values.

10.1.2.5 Liquid Crystal Display Driver Module Option

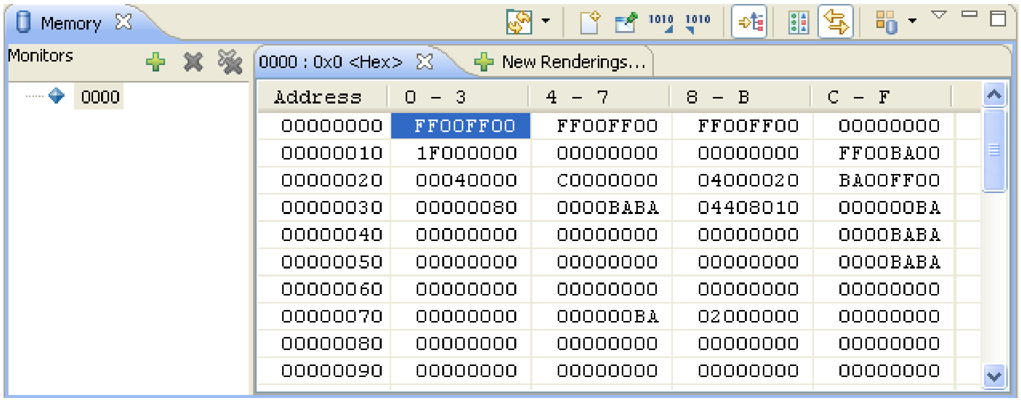
In Full Chip Simulation (FCS) Mode, this option lets you simulate all the functionality of the Liquid Crystal Display (LCD) module, including programmable LCD frame frequency, front plane pin configuration, back plane pin configuration, programmable blink frequency, and LCD interrupt flag generation. By default LCD front and back plane pins are mapped to match device use on the corresponding Freescale DEMO9RS08xx device board. These settings can be changed by you through modification of the LCDRS08V<x>_<DEVICE>.INI file, where <x> indicates the version number. This file is located in the `"<`*CW_Install*`>\prog\P&E"` folder.

10.1.2.6 Modulo Timer Interrupt Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Modulo Timer Interrupt (MTIM) Module, including:

- Programmable MTIM clock input
- Free running or modulo up count operation
- Flag polling
- Interrupt enabled mode of operation

Once the MTIM Status and Control register properly configures the operation of the module, the MTIM Counter starts incrementing. If modulo up count operation is enabled, you can observe the MTIM overflow flag in the MTIM Status and Control register in the Memory window.



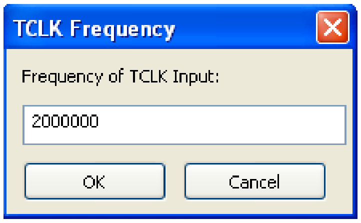
```
Address    0 - 3      4 - 7      8 - B      C - F
00000000   FF00FF00   FF00FF00   FF00FF00   00000000
00000010   1F000000   00000000   00000000   FF00BA00
00000020   00040000   C0000000   04000020   BA00FF00
00000030   00000080   0000BABA   04408010   000000BA
00000040   00000000   00000000   00000000   0000BABA
00000050   00000000   00000000   00000000   0000BABA
00000060   00000000   00000000   00000000   00000000
00000070   00000000   000000BA   02000000   00000000
00000080   00000000   00000000   00000000   00000000
00000090   00000000   00000000   00000000   00000000
```

Figure 10-16. Memory Component Window

If the MTIM interrupt is enabled, the FCS jumps to an appropriate subroutine as long as the MTIM interrupt vector is properly defined.

10.1.2.6.1 Modify MTIM TclK

The following figure shows the TclK frequency dialog box.



Frequency of TCLK Input: 2000000

Figure 10-17. TclK Frequency Dialog Box

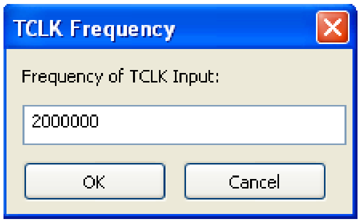
This dialog box lets you set the frequency of the TclK signal for the MTIM peripheral. In order for this value to have any effect, the TclK must be selected as the clock source for the MTIM.

10.1.2.6.2 Modulo Timer Interrupt Module User Commands

The following commands are available for the MTIM.

10.1.2.6.3 TclK Command

The TclK opens the TclK frequency dialog box shown in the following figure. You may then use this box to specify the input frequency of the TclK.



Frequency of TCLK Input: 2000000

Figure 10-18. TclK Frequency Dialog Box

Syntax

```
>gdi TclK
```

Example

```
>gdi TclK
```

Show TclK Frequency Dialog Box

10.1.2.6.4 TclK <n> Command

The TclK <n> command sets the TclK input frequency to <n>.

Syntax

```
>gdi TclK <n>
```

Where: `<n>` is the input frequency of TclK

Example

```
>gdi TclK 200000t
```

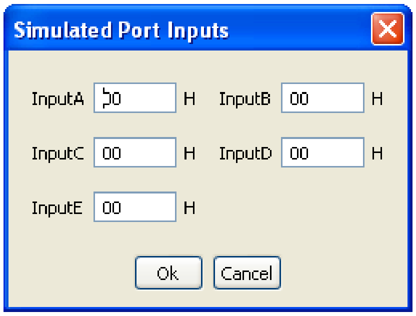
Simulate the TclK input frequency of 200000 Hz.

10.1.2.7 Input/Output (I/O) Ports Module

In Full Chip Simulation (FCS) mode, this module simulates all input and output functionality of the Input/Output (I/O) Ports module. FCS mode uses a set of designated commands to simulate the input and output activity on corresponding I/O port pins. To define an input state of the specific port, write the INPUT <x> <n> command in the Command window. The <x> represents the corresponding I/O port, while the <n> stands for the input value to write to this port. At the same time, you can use the INPUTS command to bring up the Simulated Port Inputs for all general I/O ports. It displays the current simulated values to all applicable input ports.

NOTE

Refer Input/Output Ports User Commands and IRQ Commands for more information about the various forms of this command.



InputA 00 H InputB 00 H

InputC 00 H InputD 00 H

InputE 00 H

Figure 10-19. Simulated Port Inputs Dialog Box

Use the Simulated Port Inputs dialog box to reconfigure the input value to any I/O port. Use the INPUTS command to reconfigure the output values on any relevant I/O port. You can observe the manipulation of I/O port pins in the Memory window.

Note that if the regular I/O pins are multiplexed to be used by a different MCU Module, they might not be available for general I/O functionality.

NOTE

For more information on how to properly configure I/O pins, refer to the Freescale user manual for your microprocessor.

10.1.2.7.1 Input/Output Ports User Commands

Use the following commands for general I/O ports manipulation.

10.1.2.7.2 INPUT<x> Command

The INPUT<x> command sets the simulated inputs to port <x>. The CPU reads this input value when port <x> is set as an input port.

Syntax

```
>gdi INPUT<x> <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` Eight-bit simulated value for port <x>

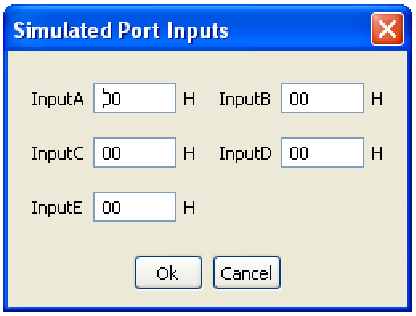
Example

```
>gdi INPUTA AA
```

Simulate the input AA on port A.

10.1.2.7.3 INPUTS Command

In FCS and CPU-Only Simulation modes, the INPUTS command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



InputA 00 H InputB 00 H

InputC 00 H InputD 00 H

InputE 00 H

Figure 10-20. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values to any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

Show I/O port input values.

10.1.2.8 Serial Communications Interface Module

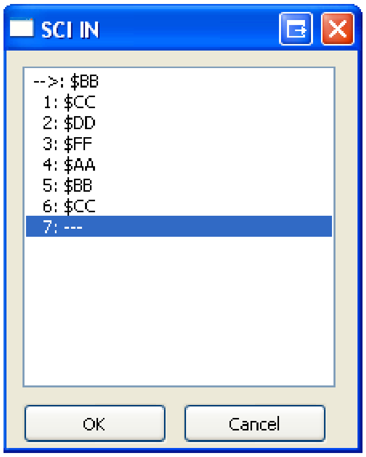
In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Serial Peripheral Interface (SPI) module including:

- Flag polling
- Interrupt enabled mode
- 8- or 9-bit length data codes
- Odd and even parity modes
- Transmission and reception of external data

FCS mode uses the buffered input/output structure to simulate SCI inputs. You can queue up to 256 data values into the input buffer. The output buffer of the SCI module can also hold 256 output values. To queue the SCI Input Data, use the SCDI <n> command in the command prompt. If <n> (the data parameter) is given, the value is placed into the next slot in the input buffer.

Otherwise, if no parameter is provided, a window is displayed with the input buffer values. You can enter input values while the window is open. An arrow points to the next value to be used as input to the SCI. The data from the SCI input buffer is written to the

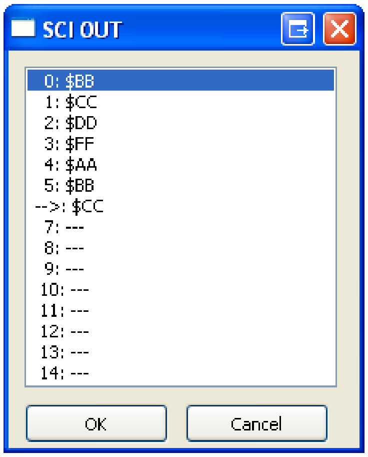
SCI data register once the SCI module has been turned on and is properly configured for receiving data from an external serial device. Once the simulation of the data transmission is over, the arrow moves to the next value in the SCI IN Buffer.



```
-->: $BB
 1: $CC
 2: $DD
 3: $FF
 4: $AA
 5: $BB
 6: $CC
 7: ---
```

Figure 10-21. SCI IN Buffer Display

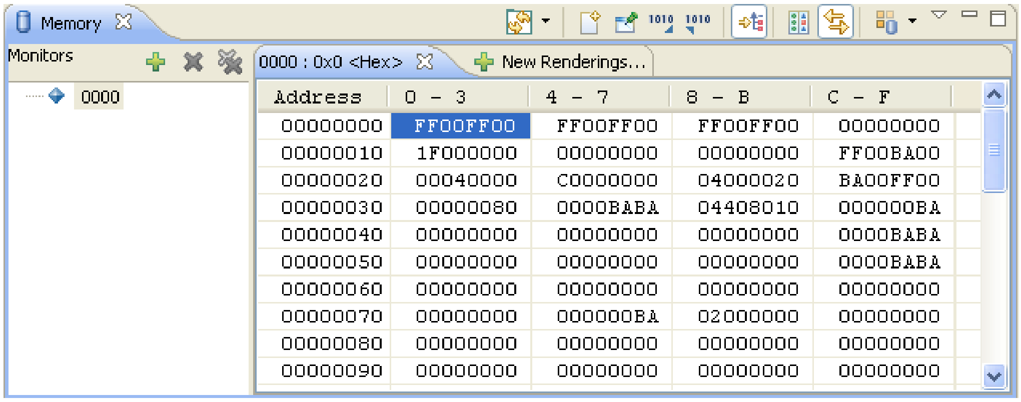
SCI Data Output Buffer simulation lets you gain access to the past 256 SCI data values transmitted out of the module. To bring up the SCI OUT buffer dialog box, use the SCDO command.



```
 0: $BB
 1: $CC
 2: $DD
 3: $FF
 4: $AA
 5: $BB
-->: $CC
 7: ---
 8: ---
 9: ---
 10: ---
 11: ---
 12: ---
 13: ---
 14: ---
```

Figure 10-22. SCI OUT Buffer Display

At any point, the SCCLR command may be used to flush the input and output SCI buffers. After the SCI simulated input is received, the first queued value is passed from the data buffer into the SCI data register. It can be observed in the memory window by displaying the memory location corresponding to the SCI data register.



```
Address    0 - 3      4 - 7      8 - B      C - F
00000000   FF00FF00   FF00FF00   FF00FF00   00000000
00000010   1F000000   00000000   00000000   FF00BA00
00000020   00040000   C0000000   04000020   BA00FF00
00000030   00000080   0000BABA   04408010   000000BA
00000040   00000000   00000000   00000000   0000BABA
00000050   00000000   00000000   00000000   0000BABA
00000060   00000000   00000000   00000000   00000000
00000070   00000000   000000BA   02000000   00000000
00000080   00000000   00000000   00000000   00000000
00000090   00000000   00000000   00000000   00000000
```

Figure 10-23. Memory Component Window

You can also observe different SCI flags in the Memory window. If the module is run in Flag Polling mode, poll the flag corresponding to the expected SCI event. If the SCI interrupts are enabled, the FCS jumps to an appropriate subroutine as long as the SCI interrupt vectors are properly defined.

NOTE

For more information on how to configure the SCI module for desired operation, refer to the Freescale user manual for your microprocessor.

10.1.2.8.1 SCI Commands

Use the following commands for serial communication interface manipulation.

10.1.2.8.2 SCCLR Command

Use the SCCLR command to flush the input and output buffers for SCI simulation. This resets the buffers and clears out all values. Note that if the SCI is in the process of shifting a value, this command allows the SCI to finish the transfer. Refer the SCDI and SCDO commands for accessing the input and output buffers of the SCI interface.

Syntax

```
>gdi SCCLR
```

Example

```
>gdi SCCLR
```

Clear input and output buffer for SCI simulation

10.1.2.8.3 SCDI Command

The SCDI command lets you input data into the SCI. If a data parameter is given, the value is placed into the next slot in the SCI input buffer. If no parameter is given, a window displays the input buffer values. Input values can be entered while the window is open. An arrow points to the next value to be used as input to the SCI. The maximum number of input values is 256 bytes.

Syntax

```
>gdi SCDI [<n>]
```

Where: `<n>` The value to be entered into the next location in the input buffer

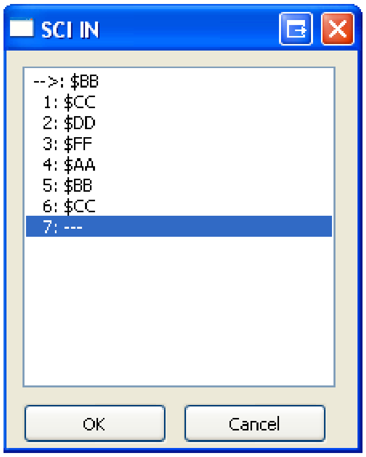
Example

```
>gdi SCDI $55
```

Set the next input value to the SCI to $55

```
>gdi SCDI
```

Pull up the data window with all the input values.



```
-->: $BB
  1: $CC
  2: $DD
  3: $FF
  4: $AA
  5: $BB
  6: $CC
  7: ---
```

Figure 10-24. SCI IN buffer display

10.1.2.8.4 SCDO Command

The SCDO command displays the output buffer from the SCI. A window is opened that shows all the data that the SCI has shifted out. An arrow points to the last output value transmitted. The maximum number of output values that the buffer holds is 256 bytes.

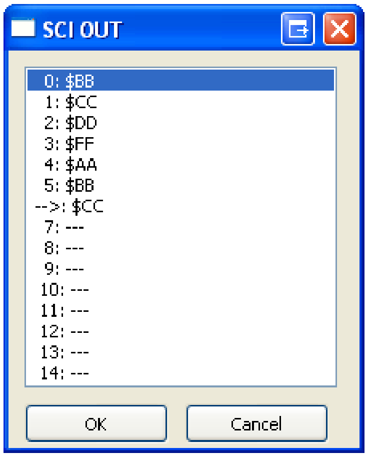
Syntax

```
>gdi SCDO
```

Example

```
>gdi SCDO
```

View data from the output buffer for the SCI simulation.



```
0: $BB
1: $CC
2: $DD
3: $FF
4: $AA
5: $BB
-->: $CC
7: ---
8: ---
9: ---
10: ---
11: ---
12: ---
13: ---
14: ---
```

Figure 10-25. SCI OUT Buffer Display

10.1.2.9 Serial Peripheral Interface Module

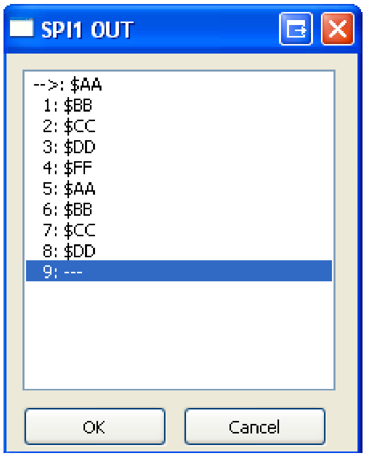
In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Serial Peripheral Interface (SPI) module including:

- Flag polling
- Interrupt enabled mode
- Master and slave modes
- Slave input clock
- Transmission and reception of external data

FCS mode uses the buffered input/output structure to simulate SPI inputs. You can queue up to 256 data values into the input buffer. The output buffer of the SPI module can also hold 256 output values. To queue the SPI Input Data, use the SPDI <n> command at the command prompt. If <n> (the data parameter) is given, the value is placed into the next slot in the input buffer.

Otherwise a window is displayed with the input buffer values. You can enter input values while the window is open. An arrow points to the next input value to the SPI. The data from the SPI input buffer is written to the SPI data register once the SPI module is turned on and is properly configured for receiving data from an external serial device. Once the simulation of the data transmission is over, the arrow moves to the next value in the SPI IN Buffer.

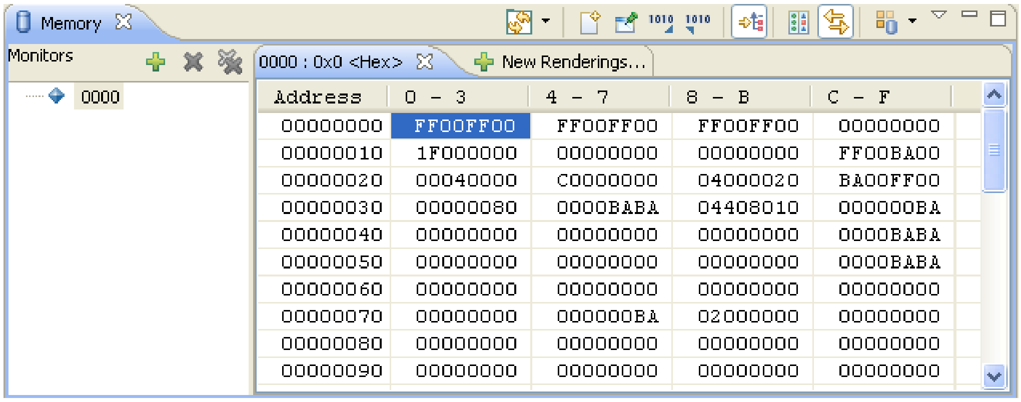
SPI data output buffer simulation lets you gain access to the past 256 SPI data values transmitted out of the module. To bring up the SPI OUT buffer dialog box, use the SPDO command.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 10-26. SPI OUT Buffer Display

The SPCLR command may be used at any point to flush the input and output SPI buffers. After the SPI simulated input is received, the first queued value is passed from the data buffer into the SPI data register. It can be observed in the Memory window by displaying the memory location corresponding to the SPI data register.



```
Address    0 - 3     4 - 7     8 - B     C - F
00000000   FF00FF00  FF00FF00  FF00FF00  00000000
00000010   1F000000  00000000  00000000  FF00BA00
00000020   00040000  C0000000  04000020  BA00FF00
00000030   00000080  0000BABA  04408010  000000BA
00000040   00000000  00000000  00000000  0000BABA
00000050   00000000  00000000  00000000  0000BABA
00000060   00000000  00000000  00000000  00000000
00000070   00000000  000000BA  02000000  00000000
00000080   00000000  00000000  00000000  00000000
00000090   00000000  00000000  00000000  00000000
```

Figure 10-27. Memory Component Window

You can also observe different SPI flags in the Memory window. If the module is run in Flag Polling mode, poll the flag corresponding to the expected SPI event. If the SPI interrupts are enabled, the FCS jumps to an appropriate subroutine as long as the SPI channel interrupt vectors are properly defined.

To simulate the frequency of the SPI slave input clock, use the SPFREQ <n> command. If the SPI is configured for slave mode, this command lets you enter the number of cycles <n> in the period of the input clock. If the SPFREQ command is not used, then clocking is set by the SPI control register.

NOTE

For more information on how to configure the SPI module for desired operation, refer to the Freescale user manual for your microprocessor.

10.1.2.9.1 SPI Commands

The following serial peripheral interface commands are available for the RS08.

10.1.2.9.2 SPCLR Command

Use the SPCLR command to flush the input and output buffers for SPI simulation. This resets the buffers and clears out all values. Notice that if the SPI is currently shifting a value, this command allows the SPI to finish the transfer. Refer the SPDI and SPDO commands for accessing the input and output buffers of the SPI interface.

Syntax

```
>gdi SPCLR
```

Example

```
>gdi SPCLR
```

Clear input and output buffer for SPI simulation

10.1.2.9.3 SPDI Command

The SPDI command lets you input data into the SPI. If a data parameter is given, the value is placed into the next slot in the SPI input buffer. If no parameter is given, a window displays the input buffer values. You can enter input values while the window is open. An arrow points to the next input value to the SPI. The maximum number of input values is 256 bytes.

Syntax

```
>gdi SPDI [<n>]
```

Where: `<n>` The value to be entered into the next location in the input buffer

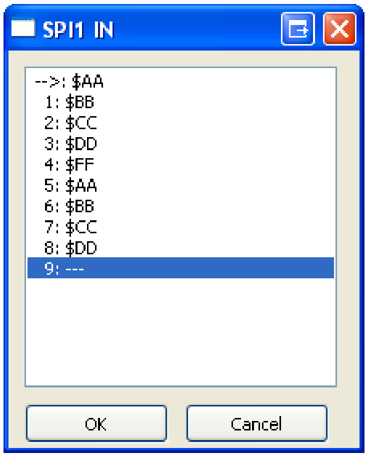
Example

```
>gdi SPDI $55
```

Set the next input value to the SPI to $55

```
>gdi SPDI
```

Pull up the data window with all the input values.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 10-28. SPI IN Buffer Display

10.1.2.9.4 SPDO Command

The SPDO command displays the output buffer from the SPI. A window opens that shows all the data that the SPI has shifted out. An arrow points to the last output value transmitted. The maximum number of output values that the buffer holds is 256 bytes.

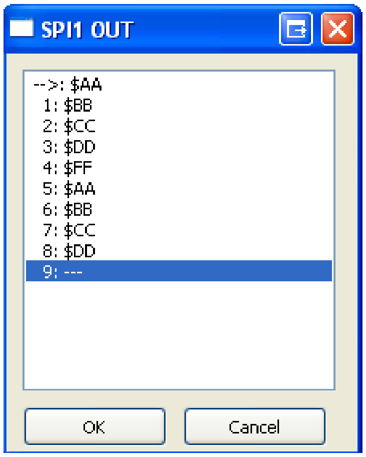
Syntax

```
>gdi SPDO
```

Example

```
>gdi SPDO
```

View data from the output buffer for the SPI simulation.



```
-->: $AA
  1: $BB
  2: $CC
  3: $DD
  4: $FF
  5: $AA
  6: $BB
  7: $CC
  8: $DD
  9: ---
```

Figure 10-29. SPI OUT Buffer Display

10.1.2.9.5 SPFREQ Command

The SPFREQ command lets you set the frequency of the SPI slave input clock. If the SPI is configured for the slave mode, this command lets you enter the number of cycles <n> per one input clock period. If no value is given, a window appears and you are prompted for a value. If this command is not used, then the clocking is assumed to be set by the SPI control register.

Syntax

```
>gdi SPFREQ [<n>]
```

Where: `<n>` The number of cycles for the period of the input clock.

Example

```
>gdi SPFREQ 8
```

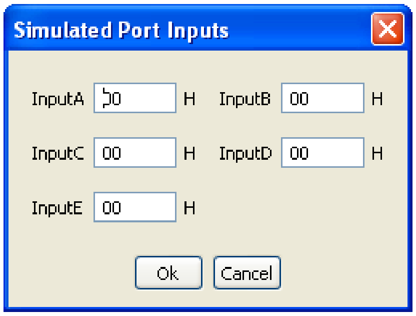
Set the period of the input slave clock to 8 cycles (total shift = 8*8 cycles per bit = 64 cycles).

10.1.2.10 Timer Interface Module

In Full Chip Simulation (FCS) mode, this module simulates all functionality of the Timer Interface module, including:

- Input capture/output compare
- Pulse width modulation
- Internal or external clock input
- Free running or modulo up count operation
- Flag polling
- Interrupt enabled mode of operation

FCS mode uses the simulated port inputs to trigger the input capture on a given timer channel. To define an input state of the specific port, use the INPUT<x> <n> command in the Command window. The <x> represents the corresponding I/O port, while <n> stands for the input value to write to this port. At the same time, you can use the INPUTS command to display the Simulated Port Inputs for all general I/O ports. It displays the current simulated values to all applicable input ports. Refer the documentation for Timer Module Commands for more information about the various forms of this command.



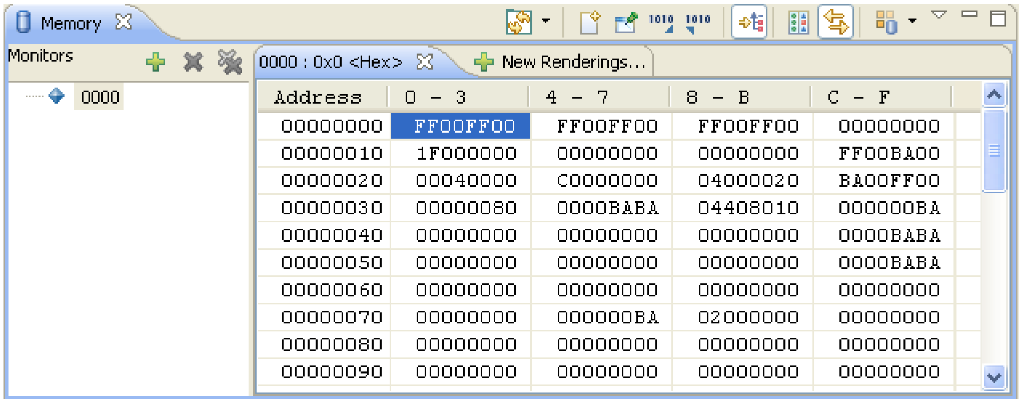
InputA 00 H InputB 00 H

InputC 00 H InputD 00 H

InputE 00 H

Figure 10-30. Simulated Port Inputs Dialog Box

Use the Simulated Port Inputs dialog box to reconfigure the input value to any I/O port. To trigger the event, first set the port inputs high or low and then invert them to an opposite value, depending on whether the input capture is set for rising/falling edge. Once the Input Capture event takes place, you can observe the CHxF in the Channel Status and Control register in the Memory window.



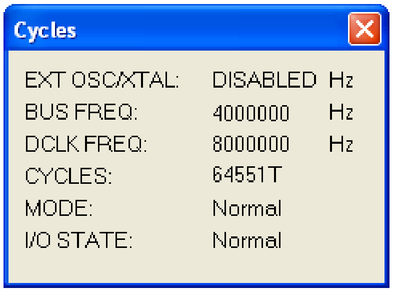
Address	0 - 3	4 - 7	8 - B	C - F
00000000	FF00FF00	FF00FF00	FF00FF00	00000000
00000010	1F000000	00000000	00000000	FF00BA00
00000020	00040000	C0000000	04000020	BA00FF00
00000030	00000080	0000BABA	04408010	000000BA
00000040	00000000	00000000	00000000	0000BABA
00000050	00000000	00000000	00000000	0000BABA
00000060	00000000	00000000	00000000	00000000
00000070	00000000	000000BA	02000000	00000000
00000080	00000000	00000000	00000000	00000000
00000090	00000000	00000000	00000000	00000000

Figure 10-31. Memory Component Window

If the Timer module is configured for an Output Compare event, then once the event takes place you can observe the same CHxF Flag via the Memory window. If the timer channel interrupt is enabled, the FCS jumps to an appropriate subroutine as long as the Timer channel interrupt vector is properly defined. To observe the Timer Overflow Flag (TOF) flag being set as a result of the corresponding CPU event, situate your Memory window on the memory location of the Timer Status and Control register.

To observe the Pulse Width Modulation (PWM) operation, properly configure the Timer to operate in the Modulo up count mode, then select the toggle-on-overflow or clear/set output on compare events to create the desired duty cycle wave. Once a PWM event takes place, you can observe pin toggle/clear/set behavior corresponding to the Timer configuration in the Memory window that is displaying the I/O port associated with a given timer channel.

To observe the accuracy of the Timer module operation, you can observe the number of CPU cycles that it takes for the event to occur. The cycle counter is only incremented as you step through the code. To determine the exact amount of cycles over which the event occurs, one can either observe the cycle display in the Register window or use the built in simulation commands. To display the current number of cycles in the Command window, use the CYCLES command. To change the number of cycles in the cycle counter, use CYCLES <n>, where <n> is the new cycle value. If the event has a pre-calculated number of cycles, use CYCLE 00 to reset the number of cycles and GOTOCYCLE <n> to run through the code until you reach the expected event.



```
EXT OSC/XTAL:   DISABLED  Hz
BUS FREQ:       4000000   Hz
DCLK FREQ:      8000000   Hz
CYCLES:         64551T
MODE:           Normal
I/O STATE:      Normal
```

Figure 10-32. Register Window With Cycles Display

10.1.2.10.1 Timer Module Commands

The following timer module commands are available for use with the HC08/HCS08 processors.

10.1.2.10.2 CYCLES Command

The CYCLES command changes the value of the cycles counter. The cycles counter counts the number of the processor cycles that have passed during execution. The Cycles Window shows the cycle counter. The cycle count can be useful for timing procedures.

Syntax

```
>gdi CYCLES <n>
```

Where: `<n>` Integer value for the cycles counter

Examples

```
>gdi CYCLES 0
```

Reset cycles counter

```
>gdi CYCLES 1000
```

Set cycle counter to 1000.

10.1.2.10.3 GOTOCYCLE Command

The GOTOCYCLE command executes the program in the simulator beginning at the address in the program counter (PC). Execution continues until the cycle counter is equal to or greater than the specified value, until a key or the Stop button on the toolbar is pressed, until it reaches a break point, or until an error occurs.

Syntax

```
>gdi GOTOCYCLE <n>
```

Where: `<n>` Cycle-counter value at which the execution stops

Example

```
>gdi GOTOCYCLE 100
```

Execute the program until the cycle counter equals 100.

10.1.2.10.4 INPUT<x> Command

The INPUT<x> command sets the simulated inputs to port <x>. The CPU reads this input value when port <x> is set as an input port.

Syntax

```
>gdi INPUT<x> <n>
```

Where: `<x>` is the letter representing corresponding port

`<n>` Eight-bit simulated value for port <x>

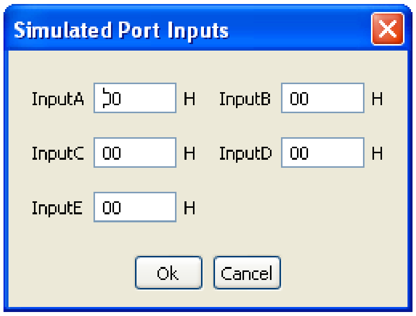
Example

```
>gdi INPUTA AA
```

Simulate the input AA on port A.

10.1.2.10.5 INPUTS Command

In FCS and CPU-Only Simulation modes, the INPUTS command opens the Simulated Port Inputs dialog box shown in the following figure. You may then use this box to specify the input states of port pins and IRQ.



Simulated Port Inputs

InputA 00 H InputB 00 H

InputC 00 H InputD 00 H

InputE 00 H

Figure 10-33. Simulated Port Inputs Dialog Box

When using In-Circuit Simulation mode, the INPUTS command shows the simulated input values to any applicable port.

Syntax

```
>gdi INPUTS
```

Example

```
>gdi INPUTS
```

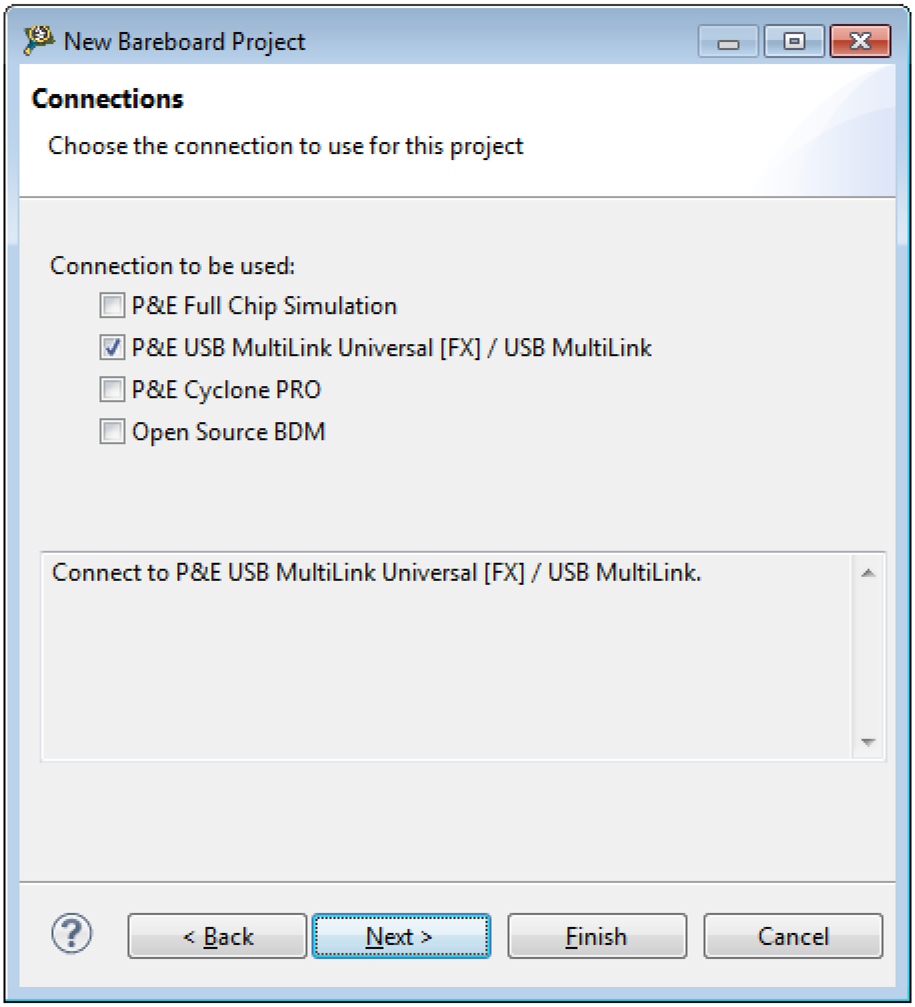
Show I/O port input values.

10.2 P&E Hardware Interface Connection for RS08

This section describes the RS08 P&E Connection options. The Connection setting permits a connection to RS08 Freescale devices via P&E Multilink, Cyclone (including the Cyclone), and OSBDM hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

10.2.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.



Connections

Choose the connection to use for this project

Connection to be used:

- □ P&E Full Chip Simulation
- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone PRO
- □ Open Source BDM

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 10-34. New project wizard

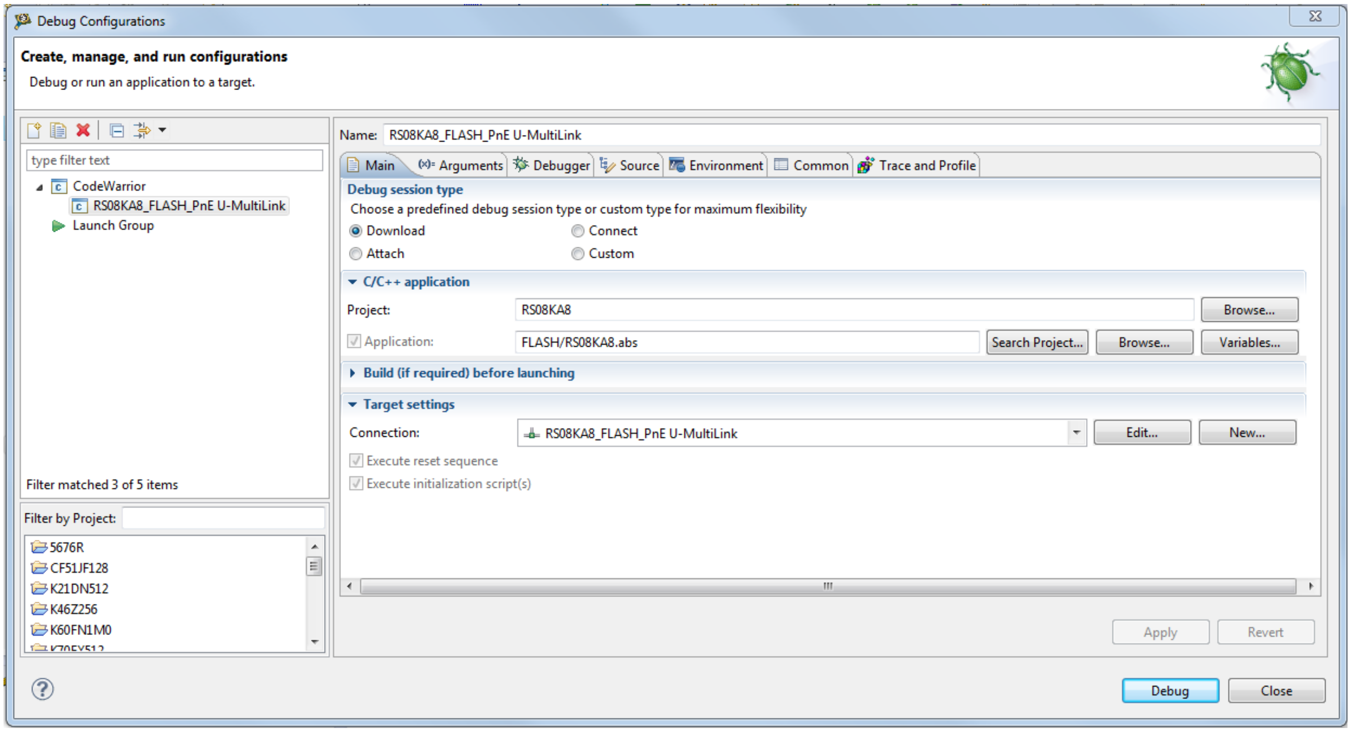
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

10.2.2 Launch Configuration Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project type you would like to set the launch configurations. Refer to the following figure.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

Name: RS08KA8_FLASH_PnE U-MultiLink

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: RS08KA8

☑ Application: FLASH/RS08KA8.abs

Build (if required) before launching

Target settings

Connection: RS08KA8_FLASH_PnE U-MultiLink

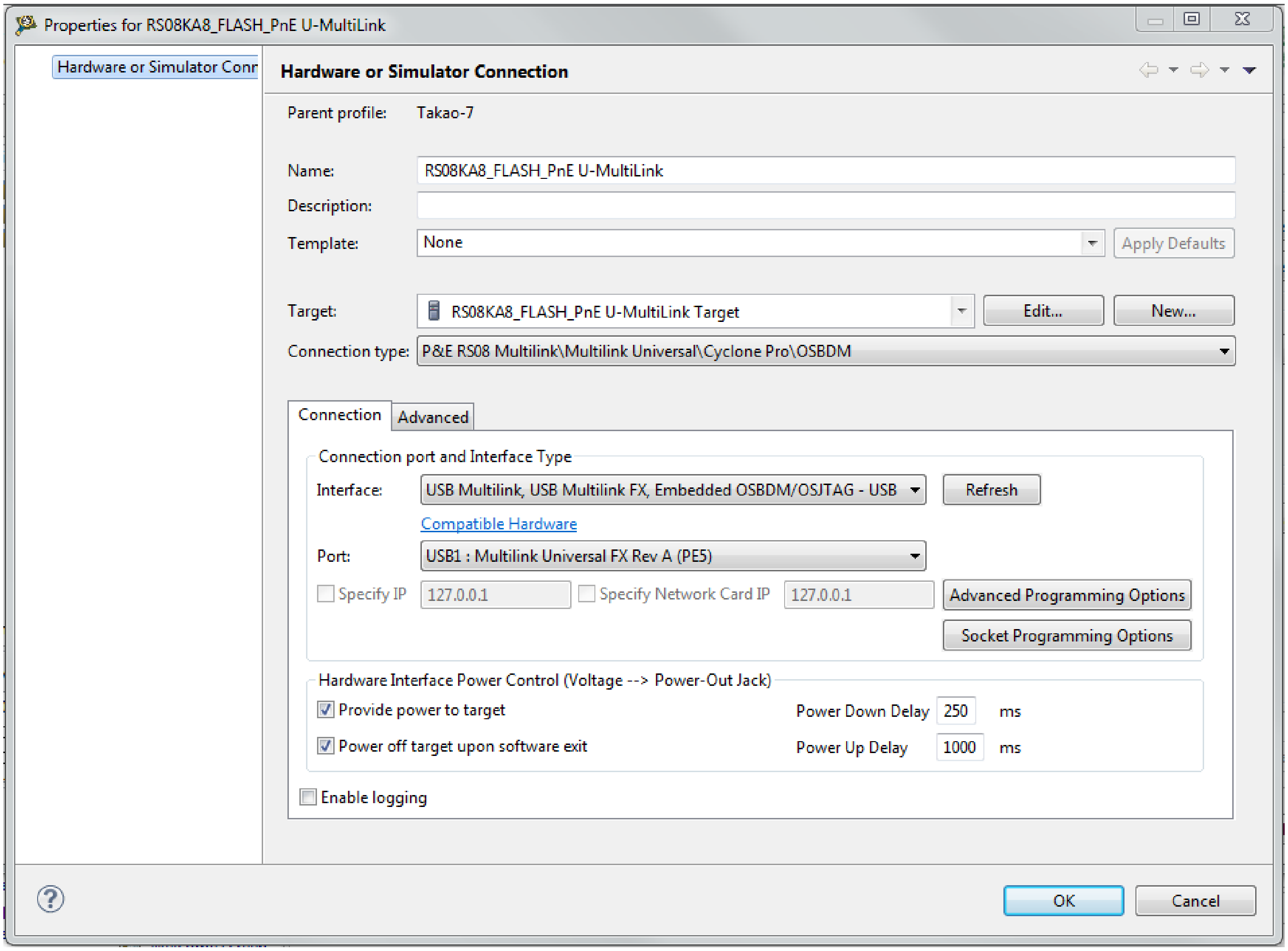
☑ Execute reset sequence

☑ Execute initialization script(s)

Filter matched 3 of 5 items

Figure 10-35. Debug Configuration Dialog Box

3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear. Refer to the following figure.



Properties for RS08KA8_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

☑ Power off target upon software exit

Power Down Delay 250 ms

Power Up Delay 1000 ms

□ Enable logging

Hardware or Simulator Conn
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
Socket Programming Options
?
OK
Cancel

Figure 10-36. Launch Configuration Dialog Box

5. Set your configurations, click the OK button, and click the Debug button to start the debugger.

10.2.3 Connection Options

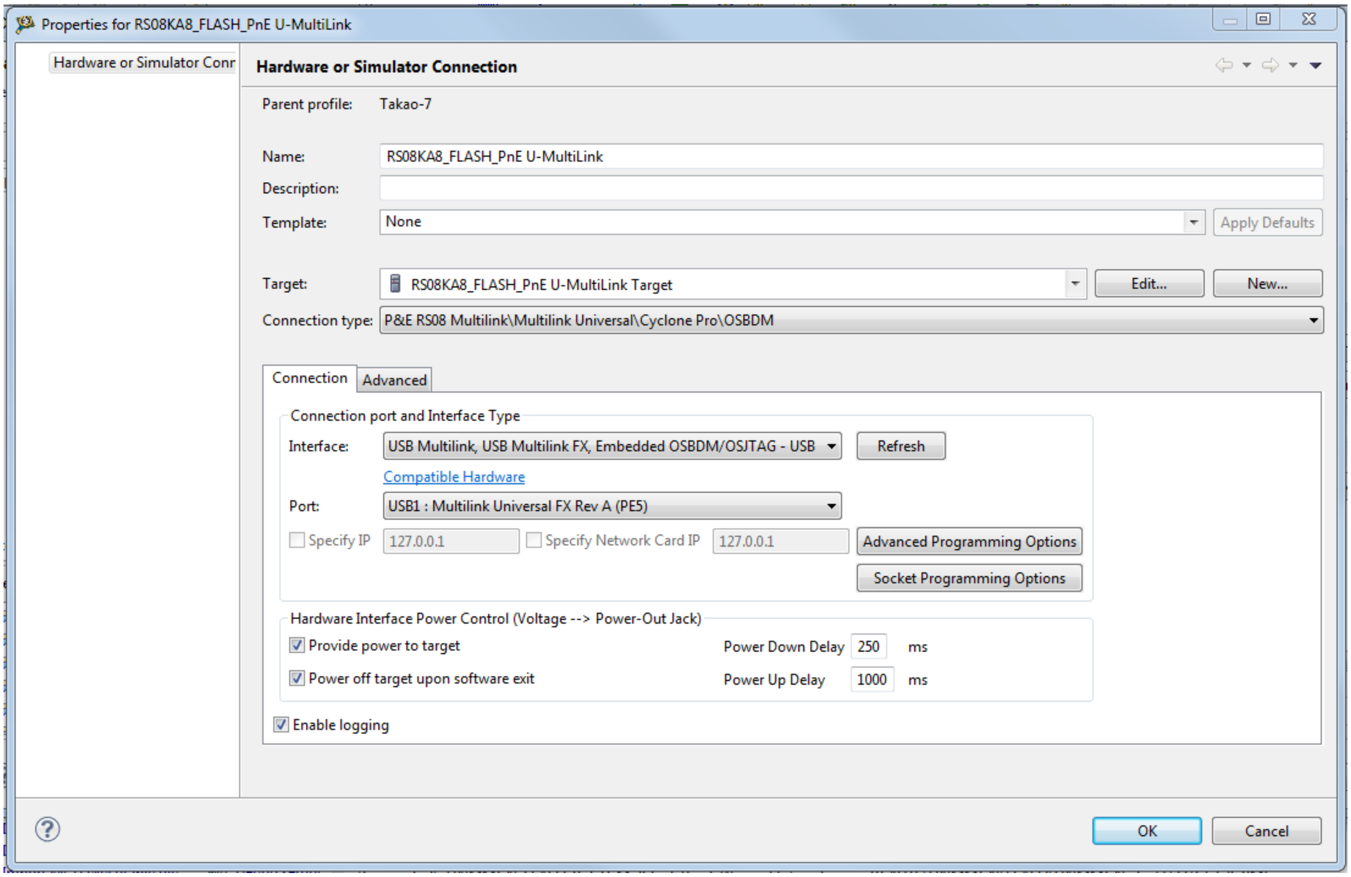
This topic describes all P&E RS08 Multilink/Cyclone/OSBDM connection options, which are common to all P&E USB Multilink Universal [FX]/USB Multilink , P&E Cyclone Serial , P&E Cyclone USB , P&E Cyclone Ethernet and Open Source BDM connections.

The options include:

- Changing P&E Connection Settings
- Connection Assistant

10.2.3.1 Changing P&E Connection Settings

All connection settings for P&E hardware interfaces are configured using the **Connection** group in the **Main** tab of the **Debugger Configuration** dialog box.



Properties for RS08KA8_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

☐ Specify IP 127.0.0.1 ☐ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

☑ Enable logging

Hardware or Simulator Conr
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
Socket Programming Options
OK
Cancel

Figure 10-37. P&E RS08 Launch Configuration Dialog Box

The following table describes the options for this view.

Table 10-1. Connection Parameter Options for P&E Multilink/Cyclone /OSBDM

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port NOTE: The USB Multilink Universal can conveniently support all Freescale architectures found in the current CodeWarrior 10 version • Cyclone - Serial Port • Cyclone - USB Port

Table continues on the next page...

Table 10-1. Connection Parameter Options for P&E Multilink/Cyclone /OSBDM (continued)

Option	Description
	• Cyclone - Ethernet Port • OSBDM NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.
Socket Programming Options	The Socket Programming Options button brings up a dialog that provides you with a graphical representation of the signals that must be connected from the BDM header to the pins of the microprocessor, in order to use Freescale socket adapters.
Advanced Programming Options	The Advanced Programming Options button brings up a dialog that provides you with options to configure the flash programming procedure.
Specify IP (Cyclone Ethernet only)	Use this option to specify the IP address of a Cyclone outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port dropdown box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Provide power to target (Cyclone and USB Multilink Universal FX only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (Cyclone and USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Regulator Output Voltage (Cyclone and USB Multilink Universal FX only)	This option adjusts the output voltage that powers the hardware target. Select a voltage value from this option's list box.
Power down delay (Cyclone and USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (Cyclone and USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.

WARNING

An improper voltage setting can damage the board.

To change P&E Connections settings, perform these steps.

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

NOTE

It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

   The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E RS08 Multilink/Cyclone/OSBDM. The P&E connections settings will appear below.
8. Click Refresh to scan valid interface and port.

   Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

NOTE

The port displayed may vary depending on the interface. For example, if you select interface as Cyclone - Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Hardware Interface Power Control (Voltage --> Power -Out Jack) group.

NOTE

This group will be enabled for the Cyclone, Tracelink and USB Multilink Universal FX interfaces only. For USB

Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

- Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
- Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.
- Select a voltage value from the Regulator Output Voltage drop-down list. This adjusts the output voltage that powers the hardware target.

NOTE

An improper voltage setting can damage the board.

- Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
- Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the hardware interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
- Click OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.
- Click Close button to close the Debug Configuration dialog box.

10.2.3.1.1 P&E Hardware Interface Connection-Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]/USB Multilink
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- Open Source BDM

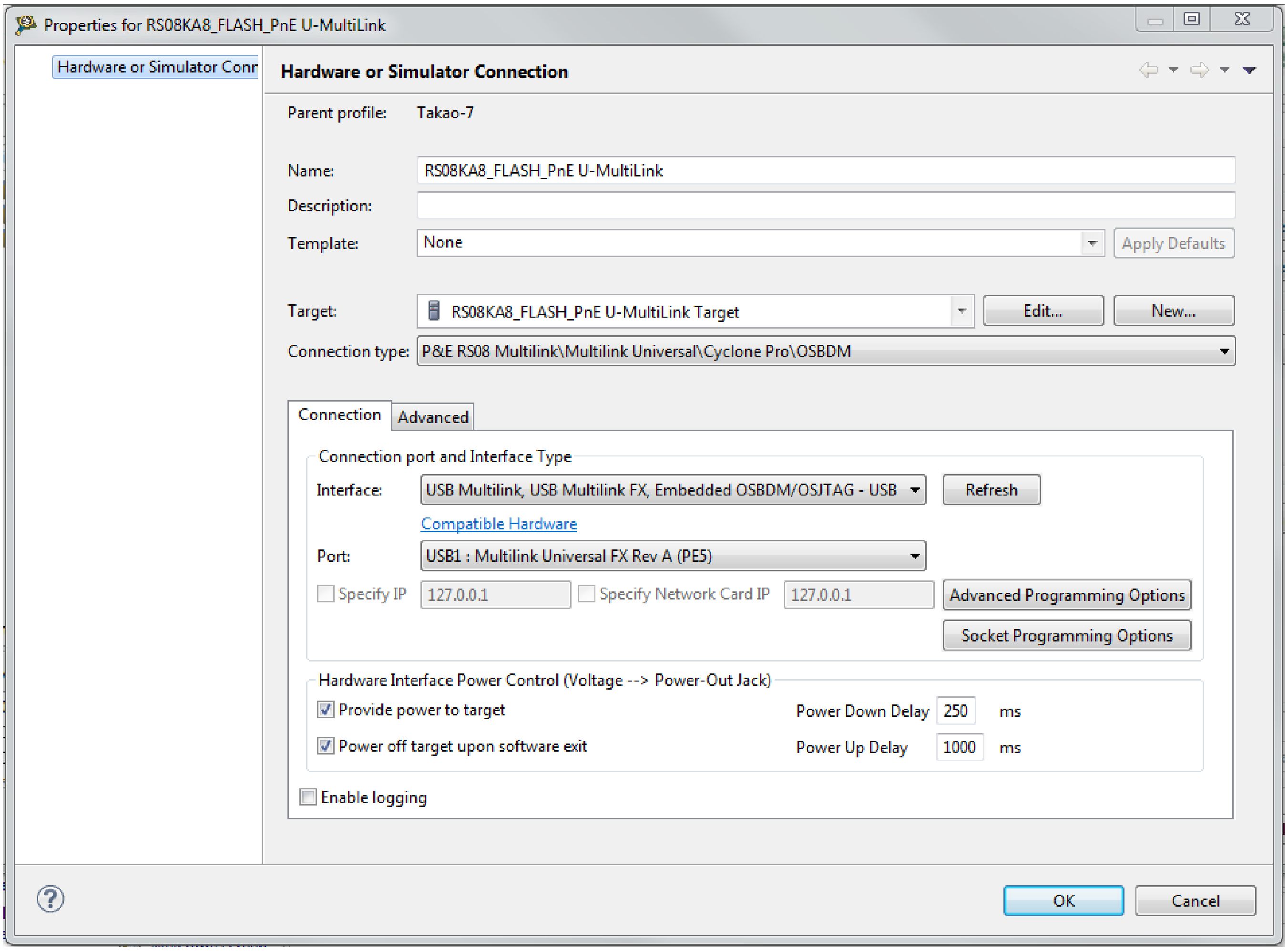
10.2.3.1.1.1 P&E USB Multilink Universal [FX]/USB Multilink

The P&E USB Multilink Universal [FX]/USB Multilink Connection setting permits a connection to USB Multilink devices, which include the P&E BDM Multilink, USB Multilink Universal, and the USB Multilink Universal FX. P&E USB Multilink Universal [FX]/USB Multilink mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

10.2.3.1.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project type you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E RS08 Multilink/Multilink Universal/Cyclone/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.


Properties for RS08KA8_FLASH_PnE U-MultiLink
Hardware or Simulator Conr
Hardware or Simulator Connection
Apply Defaults
Edit...
New...
Connection
Advanced
Connection port and Interface Type
Refresh
Compatible Hardware
Advanced Programming Options
Socket Programming Options
Hardware Interface Power Control (Voltage --> Power-Out Jack)
OK
Cancel

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

□ Enable logging

Figure 10-38. P&E's Launch Configuration Dialog Box

To use P&E's USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

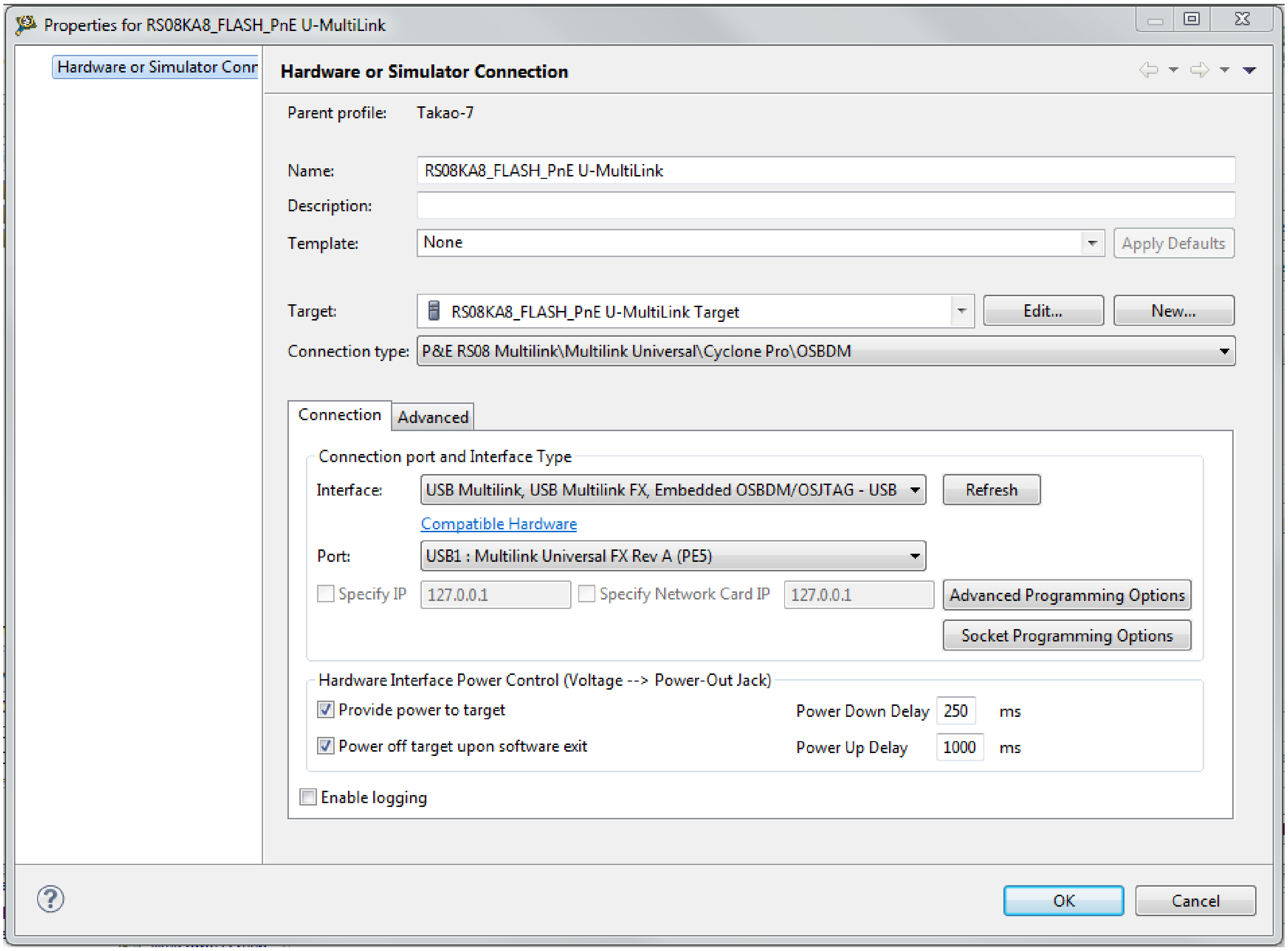
10.2.3.1.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

10.2.3.1.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E RS08 Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for RS08KA8_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

□ Enable logging

Hardware or Simulator Conn
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
Socket Programming Options
?
OK
Cancel

Figure 10-39. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone – Serial Port. The Port selection should automatically Connections — RS08 P&E Hardware Interface Connection for RS08 858

Microcontrollers V10.x Targeting Manual populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

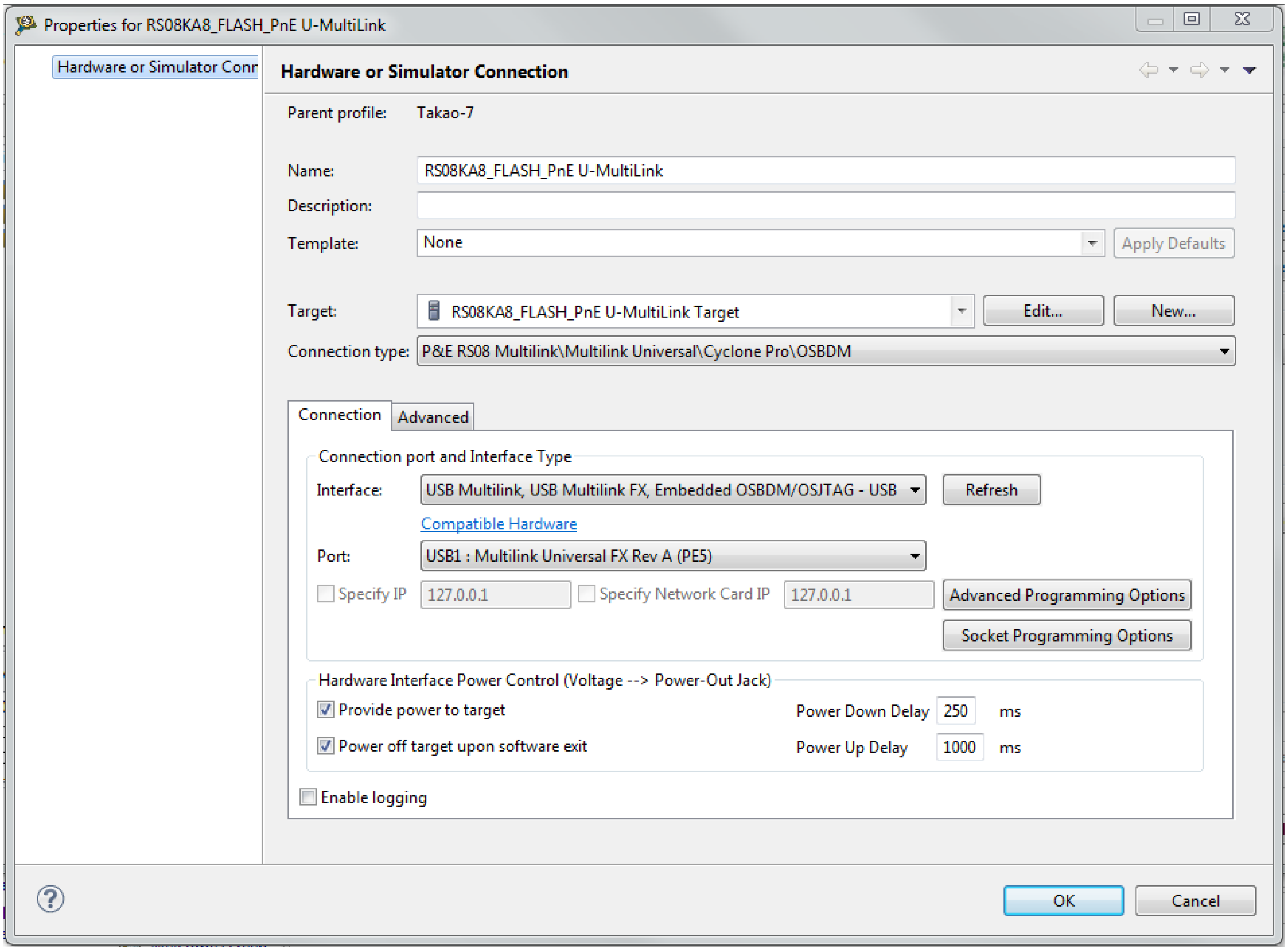
10.2.3.1.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources..

10.2.3.1.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E RS08 Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.


Properties for RS08KA8_FLASH_PnE U-MultiLink
Hardware or Simulator Conr
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
Socket Programming Options
OK
Cancel

Hardware or Simulator Connection

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

□ Enable logging

Figure 10-40. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

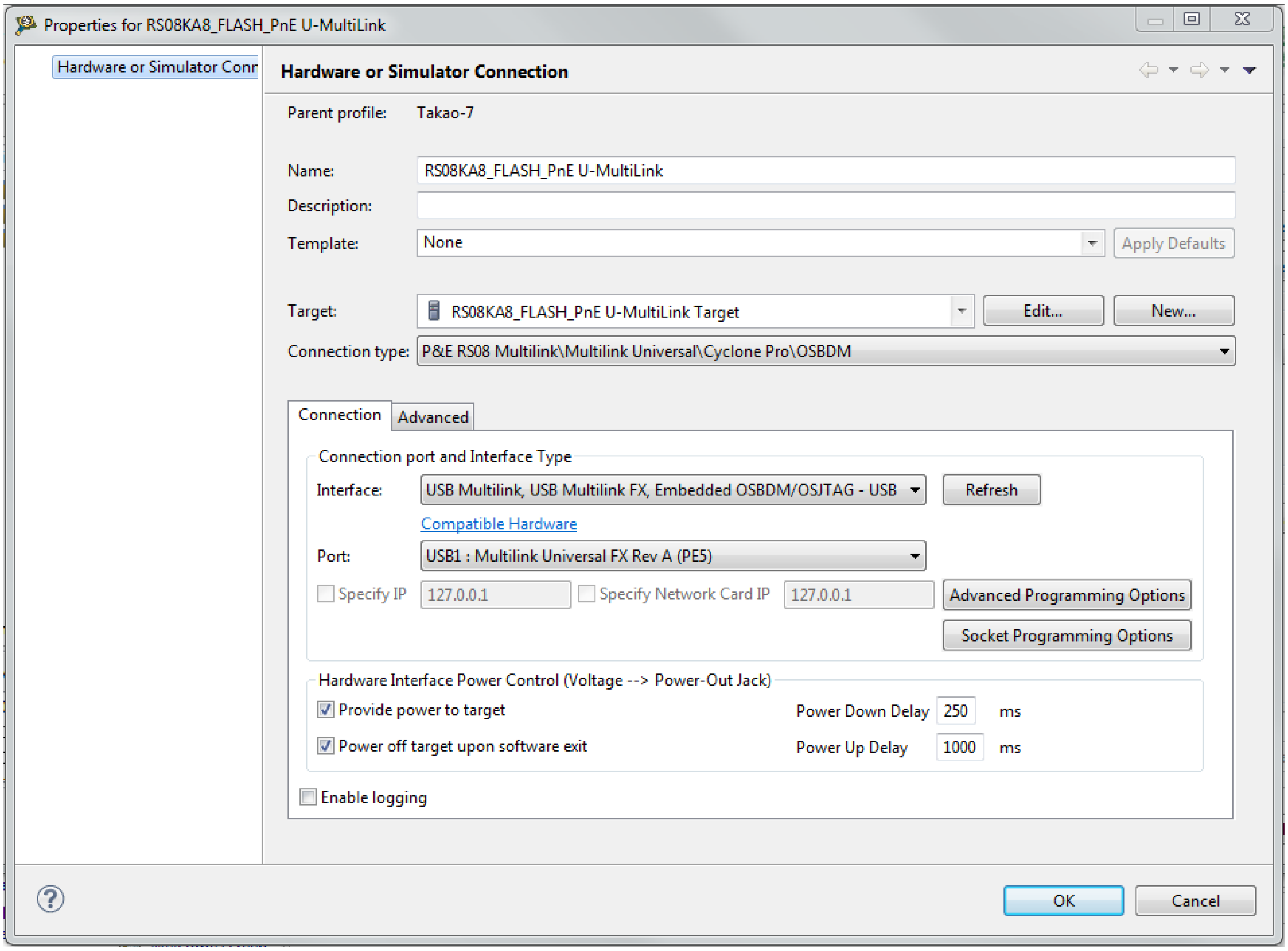
10.2.3.1.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

10.2.3.1.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E RS08 Multilink/Multilink Universal/Cyclone pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.


Properties for RS08KA8_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

□ Enable logging

Hardware or Simulator Conn
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
Socket Programming Options
OK
Cancel

Figure 10-41. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

10.2.3.1.1.5 Open Source BDM

Freescale supplies certain development boards with an integrated debug circuit based on Open Source BDM. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink. The development board also derives its power from the USB Bus.

The Open Source BDM circuit design (OSBDM-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

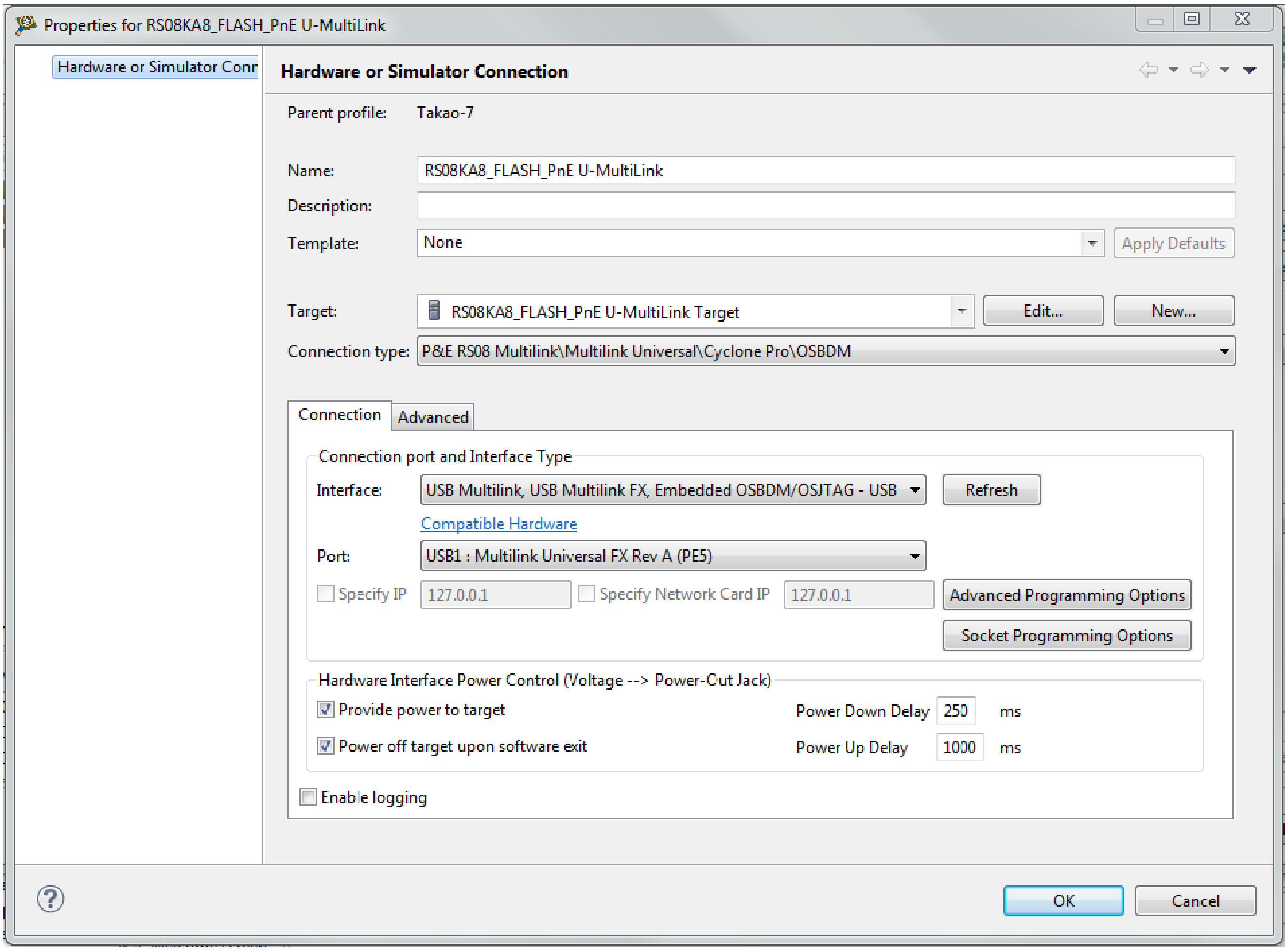
Integration with CodeWarrior is handled via the "Open Source BDM" connection. P&E has integrated the Open Source BDM support into the same connection that supports both the USB Multilink and the Cyclone. All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSBDM (albeit at a lower data rate).

The Open Source BDM Connection setting permits a connection to Open Source BDM devices. Open Source BDM mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

10.2.3.1.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E RS08 Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for RS08KA8_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: RS08KA8_FLASH_PnE U-MultiLink

Description:

Template: None

Target: RS08KA8_FLASH_PnE U-MultiLink Target

Connection type: P&E RS08 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

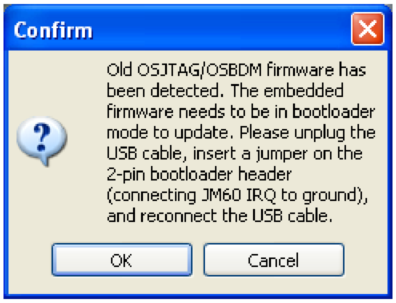
□ Enable logging

Figure 10-42. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

10.2.3.1.1.5.2 OSBDM Firmware Update

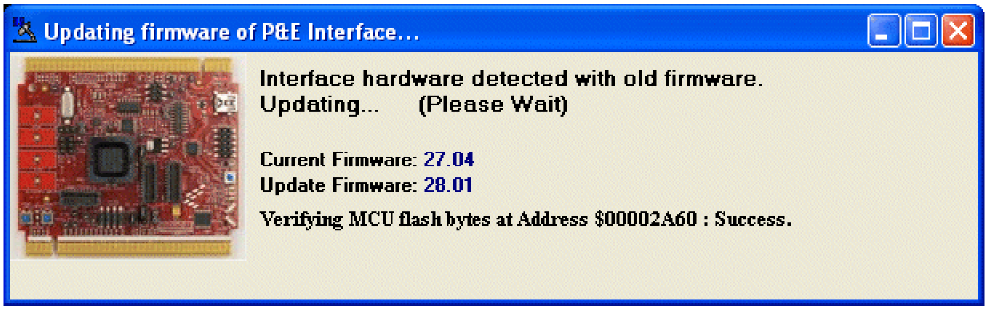
All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSBDM hardware interfaces. Whenever an OSBDM-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSBDM firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.


Confirm

Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 10-43. Old OSBDM Firmware Detected

To update the firmware, the OSBDM device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSBDM-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSBDM device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSBDM device and click on the OK button. If done correctly, the automatic firmware update will occur.


Updating firmware of P&E Interface...

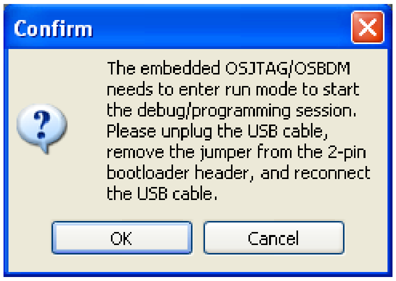
Interface hardware detected with old firmware.
Updating...   (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01

Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 10-44. OSBDM Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSBDM device must exit Bootloader mode and enter into Run mode.


Confirm

The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

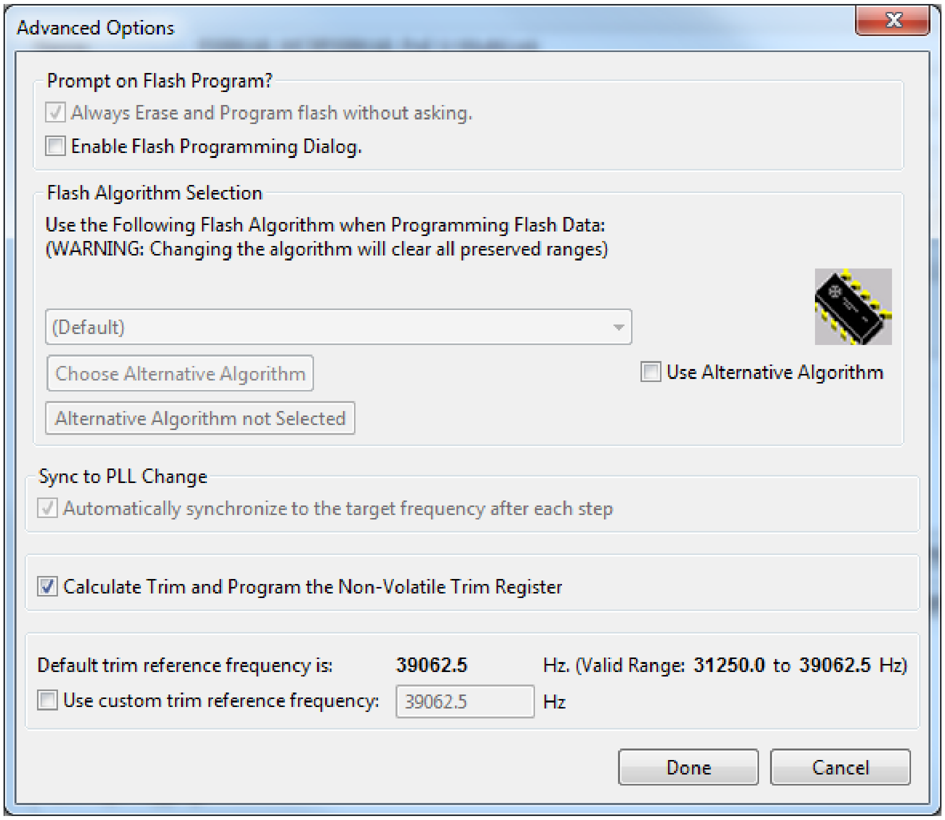
Figure 10-45. Start OSBDM Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSBDM device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSBDM firmware. If for any reason you experience difficulty performing OSBDM firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

10.2.3.1.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the Flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

□ Use Alternative Algorithm

Alternative Algorithm not Selected

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

☑ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: 39062.5 Hz. (Valid Range: 31250.0 to 39062.5 Hz)

□ Use custom trim reference frequency: 39062.5 Hz

Figure 10-46. Advanced Options Dialog Box

10.2.3.1.2.1 Enable Flash Programming Dialog Box

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

10.2.3.1.2.1.1 Trim Options

The Calculate Trim and Program the Non-Volatile Trim Register checkbox enables automatic calculation and programming of the trim value to a designated Non-Volatile memory location.

10.2.3.1.2.1.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the entire EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase and re-programmed immediately afterwards, thereby preserving the data in these ranges. Any attempts to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

10.2.3.1.2.1.3 Sync to PLL Change Checkbox

The debugger requires the Sync to PLL Change to synchronize the software/hardware connection with the microprocessor during the Flash erasing/programming procedure.

10.2.3.1.2.1.4 Calculate and Program Non-Volatile Trim

The checkbox gives you the option of trimming device to default center frequency. If this checkbox is selected, a calculated trim frequency will be programmed to a dedicated non-volatile memory location during the next debugging session.

10.2.3.1.2.1.5 Custom Trim

When the checkbox is checked, you have the ability to input a custom center frequency within an allowed range for a given device. A trim value based on this frequency will be calculated and programming into dedicated non-volatile memory location during the next debug session.

NOTE

For more information about the specific functionality of the internal clock source, see the Freescale Data Sheet for your specific device.

10.2.3.1.2.1.6 Alternative Algorithm Functionality

Once you create a project for a specific HCS08/RS08/CFV1 microprocessor, the debugger specifies a default algorithm to use during all Flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the `<CW_Install>/MCU/bin/plugins/support/HC08/gdi/P&E` directory.

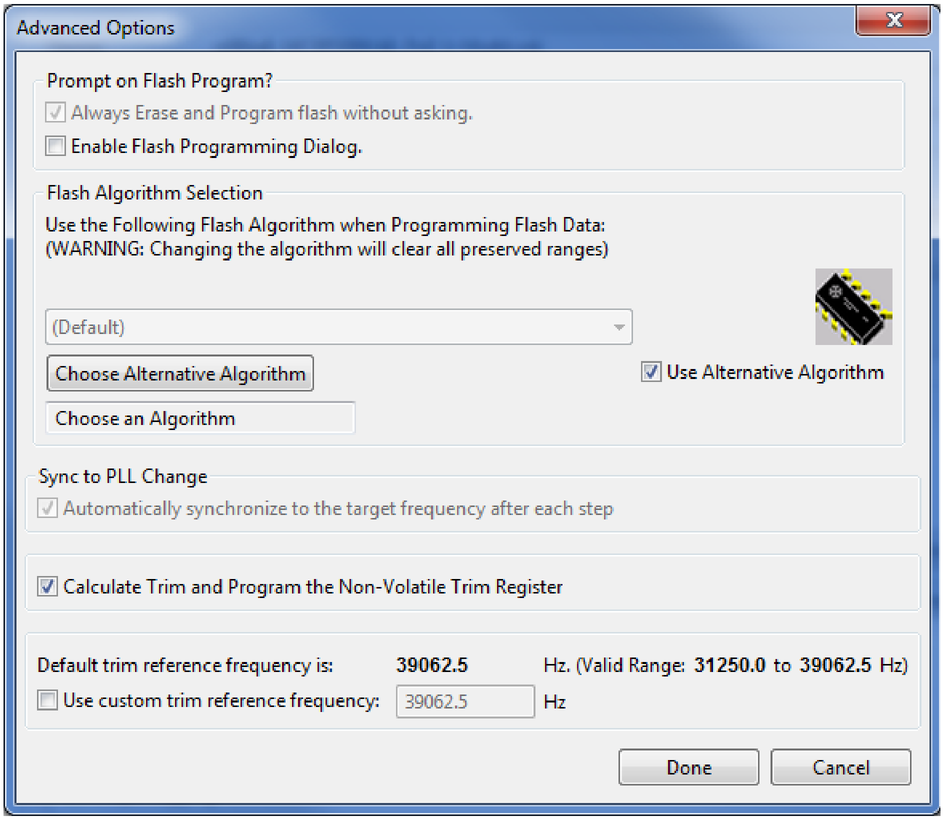
However, the default algorithm may be overridden via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. You can use this feature to select a custom programming algorithm, or simply select another one of P&E's many programming algorithms for use with a specific project.

CAUTION

> Selecting the wrong programming algorithm may damage their device, lead to under/over programming situations, or simply not program portions of the project file. You are recommended to use the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. Check the Use Alternative Algorithm checkbox.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

☑ Use Alternative Algorithm

Choose an Algorithm

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

☑ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: **39062.5** Hz. (Valid Range: **31250.0** to **39062.5** Hz)

□ Use custom trim reference frequency: 39062.5 Hz

Figure 10-47. Advanced Options - Alternative Algorithm Checkbox

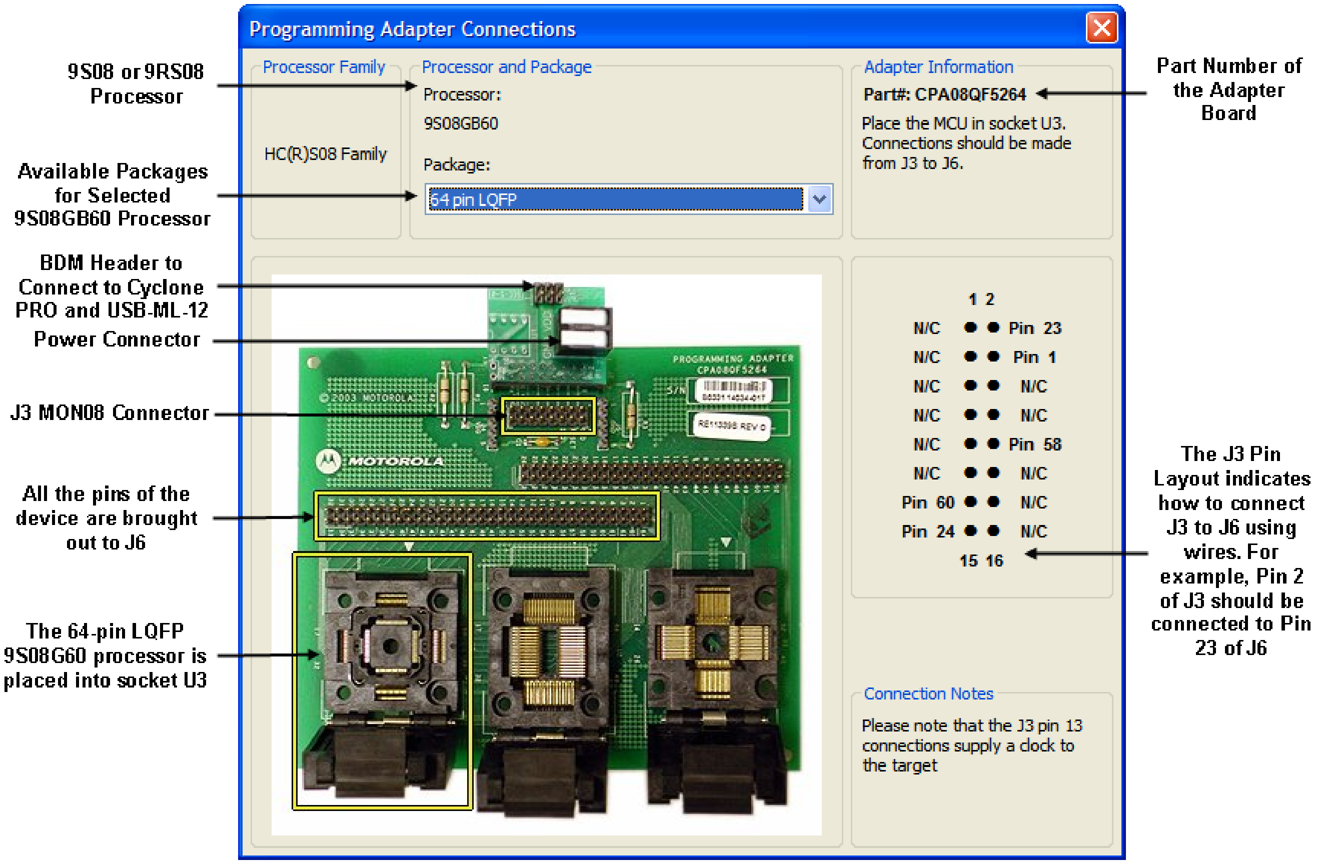
2. Click the **Choose Alternative Algorithm** button, which lets you browse for an alternative algorithm.
3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the **Choose Alternative Algorithm** button.

At this point, the current project performs all future Flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

10.2.3.1.3 Socket Programming Options Button

The Programming Adapter Connections dialog assistant is designed to facilitate the use of an extensive set of Freescale programming socket adapters. This dialog can be used to get a graphical representation of the signals that must be connected from the BDM header to the pins of the microprocessor. Making these connections lets you establish communication with a given device via a hardware debug interface.

The Socket Programming Options button in the BDM Launch Configuration dialog box takes you to the Programming Adapter Connections dialog box (see the following figure), where you can look up pin connection settings for the selected package type of the target processor. Only available package types for each target processor are listed in the Package drop-down menu. Once you have selected a package type, the Adapter Information section provides the part number of the adapter board, the socket number where the processor should be placed, and a pair of header numbers that indicate which connections should be made between them. Immediately below the Adapter Information section you will find a pin layout that displays the required connections between the aforementioned pair of headers.



Programming Adapter Connections

Processor Family

HC(R)S08 Family

Processor and Package

Processor:

9S08GB60

Package:

64 pin LQFP

Adapter Information

Part#: CPA08QF5264

Place the MCU in socket U3. Connections should be made from J3 to J6.

	1	2	
N/C	●	●	Pin 23
N/C	●	●	Pin 1
N/C	●	●	N/C
N/C	●	●	N/C
N/C	●	●	Pin 58
N/C	●	●	N/C
Pin 60	●	●	N/C
Pin 24	●	●	N/C
	15	16	

Connection Notes

Please note that the J3 pin 13 connections supply a clock to the target

9S08 or 9RS08 Processor

Available Packages for Selected 9S08GB60 Processor

BDM Header to Connect to Cyclone PRO and USB-ML-12

Power Connector

J3 MON08 Connector

All the pins of the device are brought out to J6

The 64-pin LQFP 9S08G60 processor is placed into socket U3

Part Number of the Adapter Board

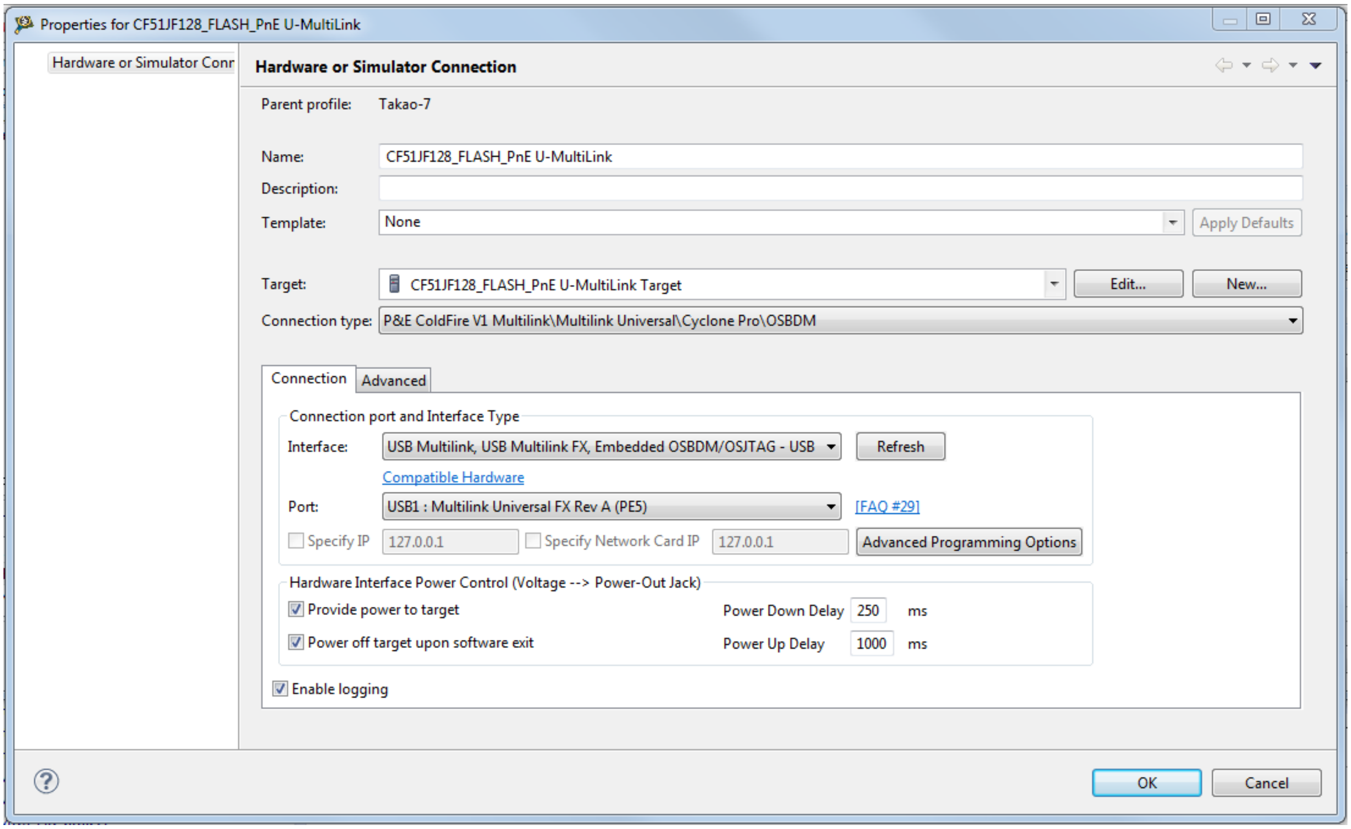
The J3 Pin Layout indicates how to connect J3 to J6 using wires. For example, Pin 2 of J3 should be connected to Pin 23 of J6

Figure 10-48. Programming Adapter Connections Dialog Box

10.2.3.2 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug and the program cannot connect to the interface hardware specified in the Launch Configuration dialog box. To edit or change your debugger connection:

1. Select the P&E device that you are using from the first drop-down menu and click **Refresh**.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the Cyclone Power Control panel to configure the power and delay settings (Cyclone only). Refer to Connection Parameter Options for P&E RS08 Multilink/ Cyclone /OSBDM for more details of each setting.
4. Click the Retry button



Properties for CF51JF128_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: CF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: CF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

☑ Power off target upon software exit

Power Down Delay 250 ms

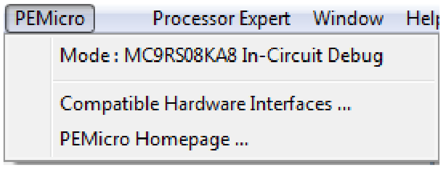
Power Up Delay 1000 ms

☑ Enable logging

Figure 10-49. Connection Assistant Interface Selected

10.2.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.



PEMicro
Processor Expert
Window
Hel
Mode : MC9RS08KA8 In-Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 10-50. Additional Connection Menu Options

Chapter 11
Connections - ColdFire V1/ColdFire+ V1

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the ColdFire V1/ColdFire+ V1 target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with the ColdFire V1/ColdFire+ V1 device family.

NOTE

The MMA95xx sensor chip uses the ColdFire V1 chip core; therefore, users should use the ColdFire V1/ColdFire+ V1 connection settings within the CodeWarrior debugger.

11.1 P&E Hardware Interface Connections for ColdFire V1

This section describes the ColdFire V1/ColdFire+ V1 P&E Connection options. The Connection setting permits a connection to CFV1/CF+V1 Freescale devices via P&E Multilink, Cyclone, and OSBDM hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

This topic describes all P&E Multilink/Cyclone/OSBDM connection options, which are common to all P&E Cyclone Serial, P&E Cyclone USB, P&E Cyclone Ethernet, and Open Source BDM connections.

The options include:

- New Project Wizard
- Launch Configurations Settings
- Changing P&E Connection Settings

11.1.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.

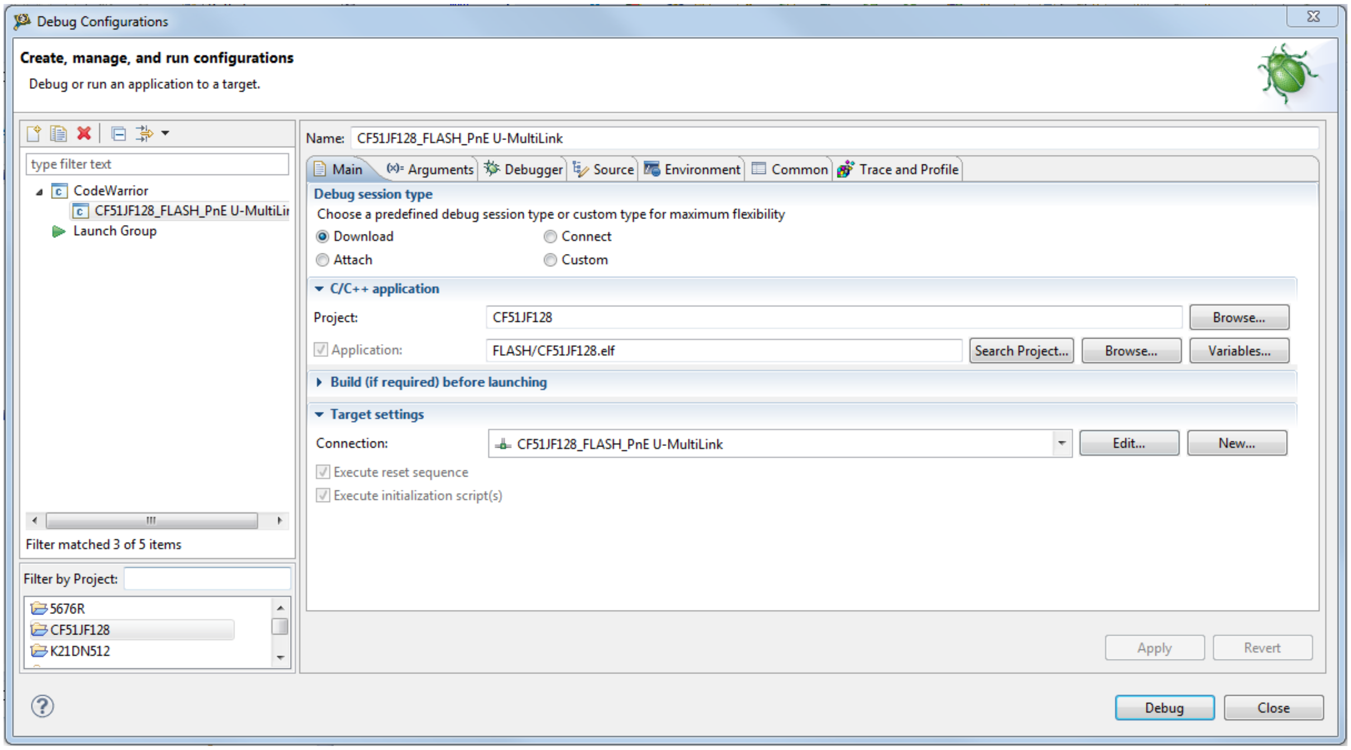
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

11.1.2 Launch Configurations Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project type you would like to set the launch configurations. Refer to the following figure.
3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear.
5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Create, manage, and run configurations

Debug or run an application to a target.

Filter matched 3 of 5 items

Filter by Project:

- 5676R
- CF51JF128
- K21DN512

Name: CF51JF128_FLASH_PnE U-MultiLink

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: CF51JF128

☑ Application: FLASH/CF51JF128.elf

Build (if required) before launching

Target settings

Connection: CF51JF128_FLASH_PnE U-MultiLink

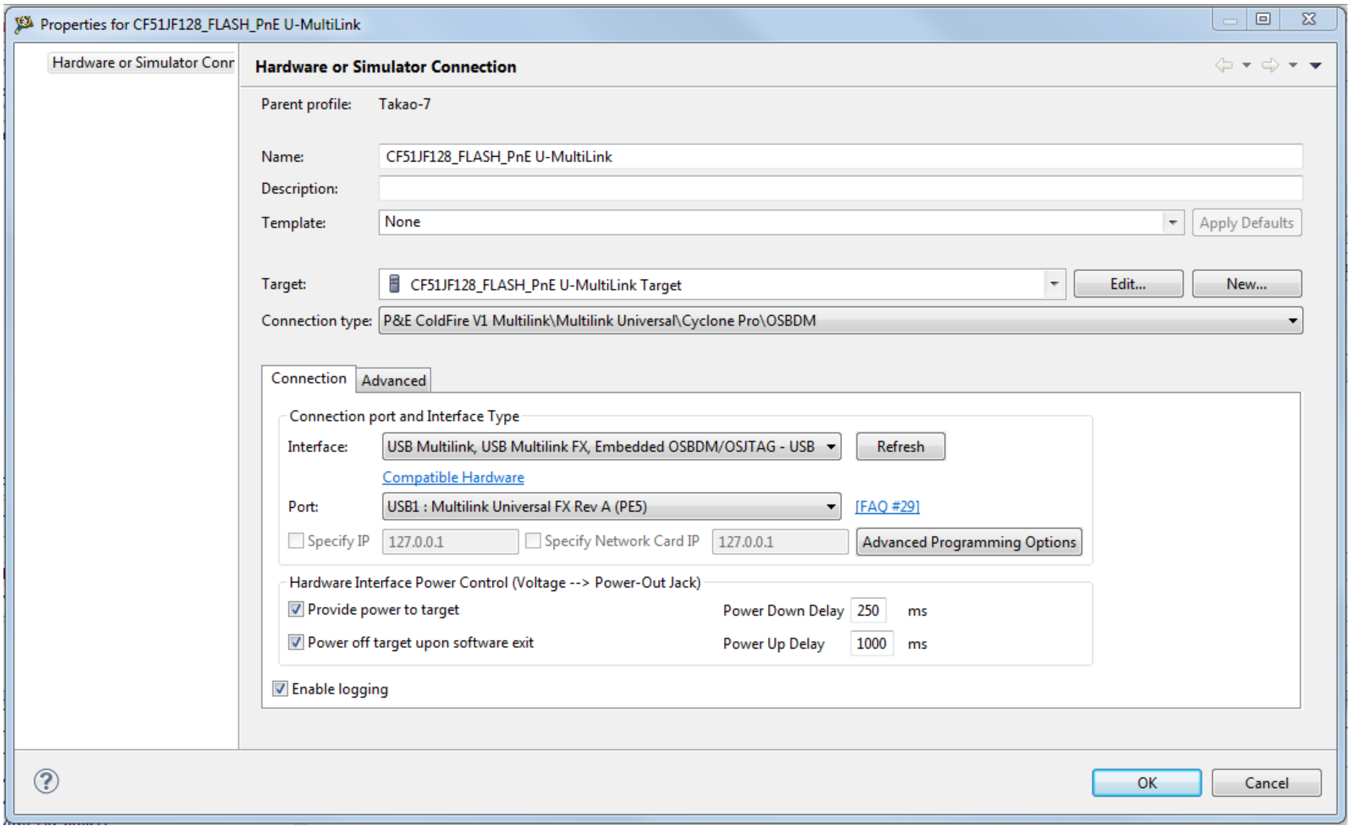
☑ Execute reset sequence

☑ Execute initialization script(s)

Figure 11-1. Debug Configuration Dialog Box

11.1.3 Changing P&E Connection Settings

All connection settings for P&E hardware interfaces are configured in the Launch Configurations dialog box.



Properties for CF51JF128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: CF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: CF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1

□ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

☑ Power off target upon software exit

Power Down Delay 250 ms

Power Up Delay 1000 ms

☑ Enable logging

Figure 11-2. P&E ColdFire V1 Launch Configuration Dialog Box

The following table describes the options for this dialog box.

Table 11-1. Connection Parameter Options for P&E Multilink /Cyclone/OSBDM

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port • Cyclone - Serial Port • Cyclone - USB Port • Cyclone - Ethernet Port NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.

Table continues on the next page...

Table 11-1. Connection Parameter Options for P&E Multilink /Cyclone/OSBDM (continued)

Option	Description
Socket Programming Options	The Socket Programming Options button brings up a dialog that provides you a graphical representation of the signals that must be connected from the BDM header to the pins of the microprocessor, in order to use Freescale socket adapters.
Advanced Programming Options	The Advanced Programming Options button brings up a dialog that provides you with options to configure the flash programming procedure.
Specify IP (Cyclone Ethernet only)	Use this option to specify the IP address of a Cyclone outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port dropdown box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Provide power to target (Cyclone and USB Multilink Universal FX only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (Cyclone and USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Regulator Output Voltage (Cyclone and USB Multilink Universal FX only)	This option adjusts the output voltage that powers the hardware target. Select a voltage value from this option's list box. NOTE: An improper voltage setting can damage the board.
Power down delay (Cyclone and USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (Cyclone and USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.

WARNING

An improper voltage setting can damage the board.

To change P&E Connections settings, perform these steps.

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

NOTE

It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E ColdFire V1 Multilink/Cyclone/OSBDM. The P&E connections settings will appear below.
8. Click Refresh to scan valid interface and port.

   Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

NOTE

The port displayed may vary depending on the interface. For example, if you select interface as Cyclone- Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Hardware Interface Power Control (Voltage --> Power -Out Jack) group.

NOTE

This group will be enabled for the Tracelink and USB Multilink Universal FX interfaces only. For USB Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

- Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
- Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.

- Select a voltage value from the Regulator Output Voltage drop-down list. This adjusts the output voltage that powers the hardware target.

WARNING
An improper voltage setting can damage the board.

- Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
- Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the hardware interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
- Click OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.
- Click Close button to close the Debug Configuration dialog box.

11.1.3.1 P&E Hardware Interface Connection-Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]/USB Multilink
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- Open Source BDM

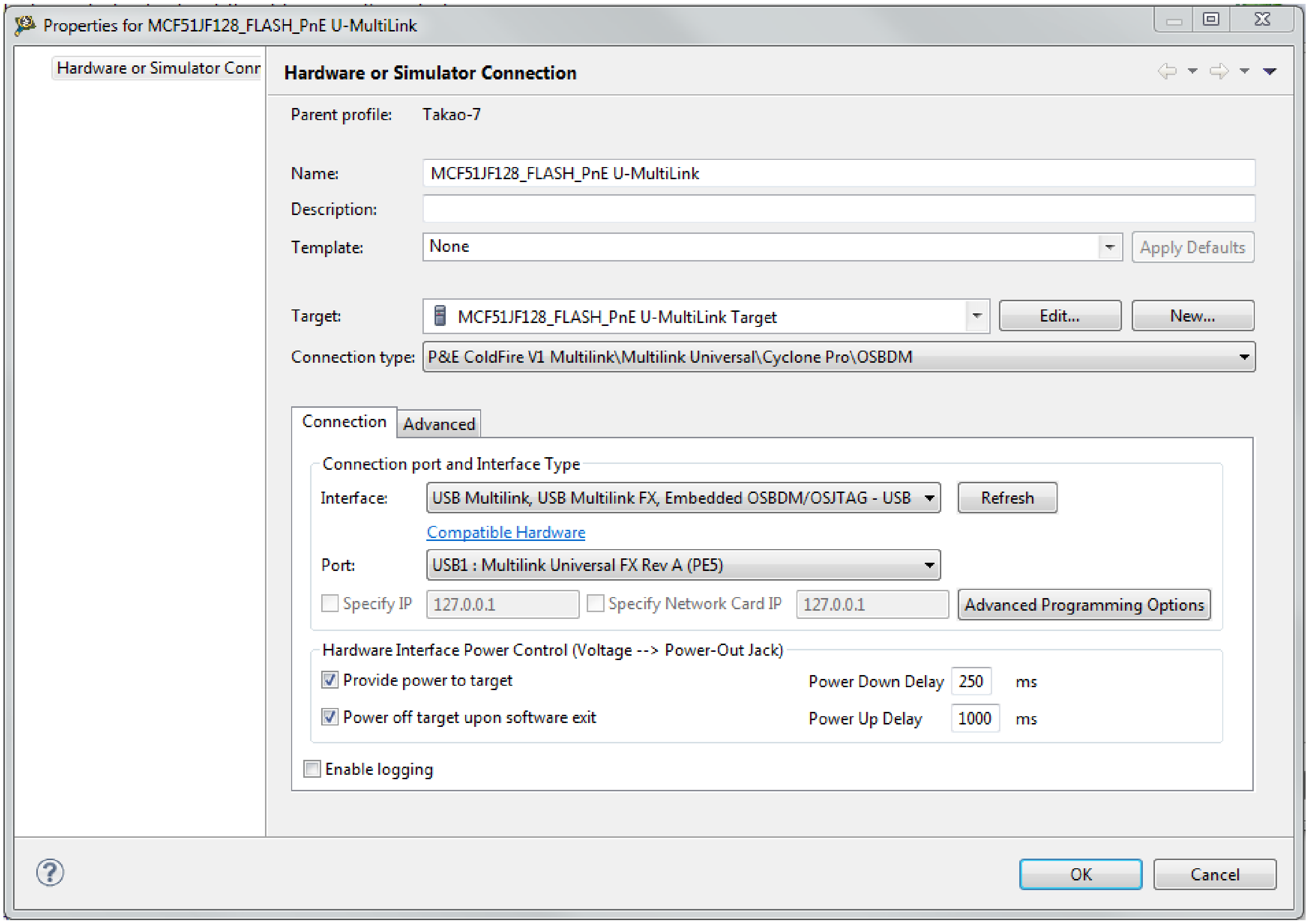
11.1.3.1.1 P&E USB Multilink Universal [FX]/USB Multilink

The P&E USB Multilink Universal [FX]/ USB Multilink Connection setting permits a connection to USB Multilink devices, which include the P&E BDM Multilink, USB Multilink Universal, and the USB Multilink Universal FX. P&E USB Multilink Universal [FX]/USB Multilink mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

11.1.3.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V1 Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.


Properties for MCF51JF128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

☑ Power off target upon software exit

Power Down Delay 250 ms

Power Up Delay 1000 ms

□ Enable logging

Figure 11-3. P&E's Launch Configuration Dialog Box

To use P&E's USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

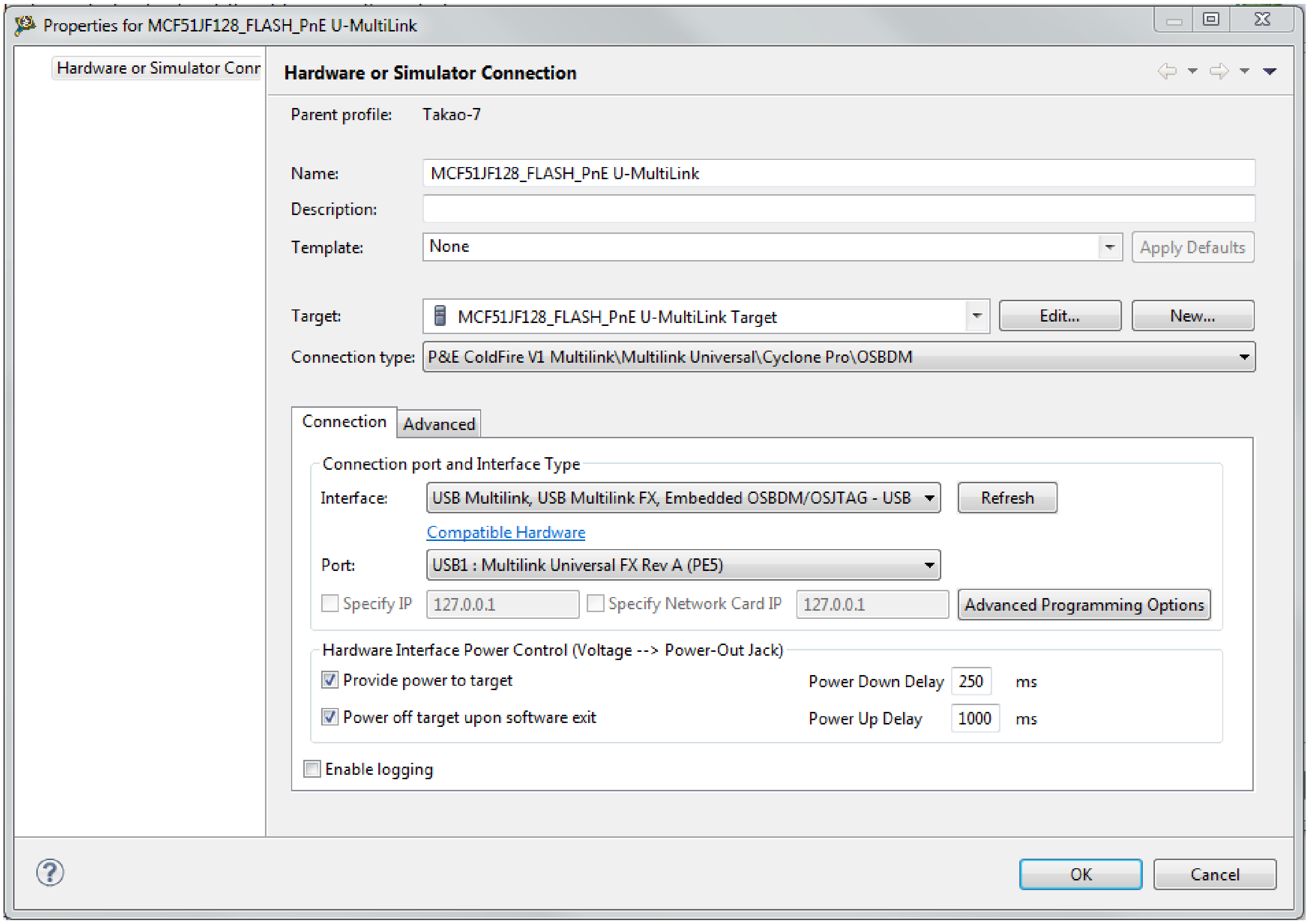
11.1.3.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

11.1.3.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V1 Multilink/Multilink Universal/Cyclone Pro/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.


Properties for MCF51JF128_FLASH_PnE U-MultiLink
Hardware or Simulator Conr
Hardware or Simulator Connection
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
OK
Cancel

Parent profile: Takao-7

Name: MCF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

□ Enable logging

Figure 11-4. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone - Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

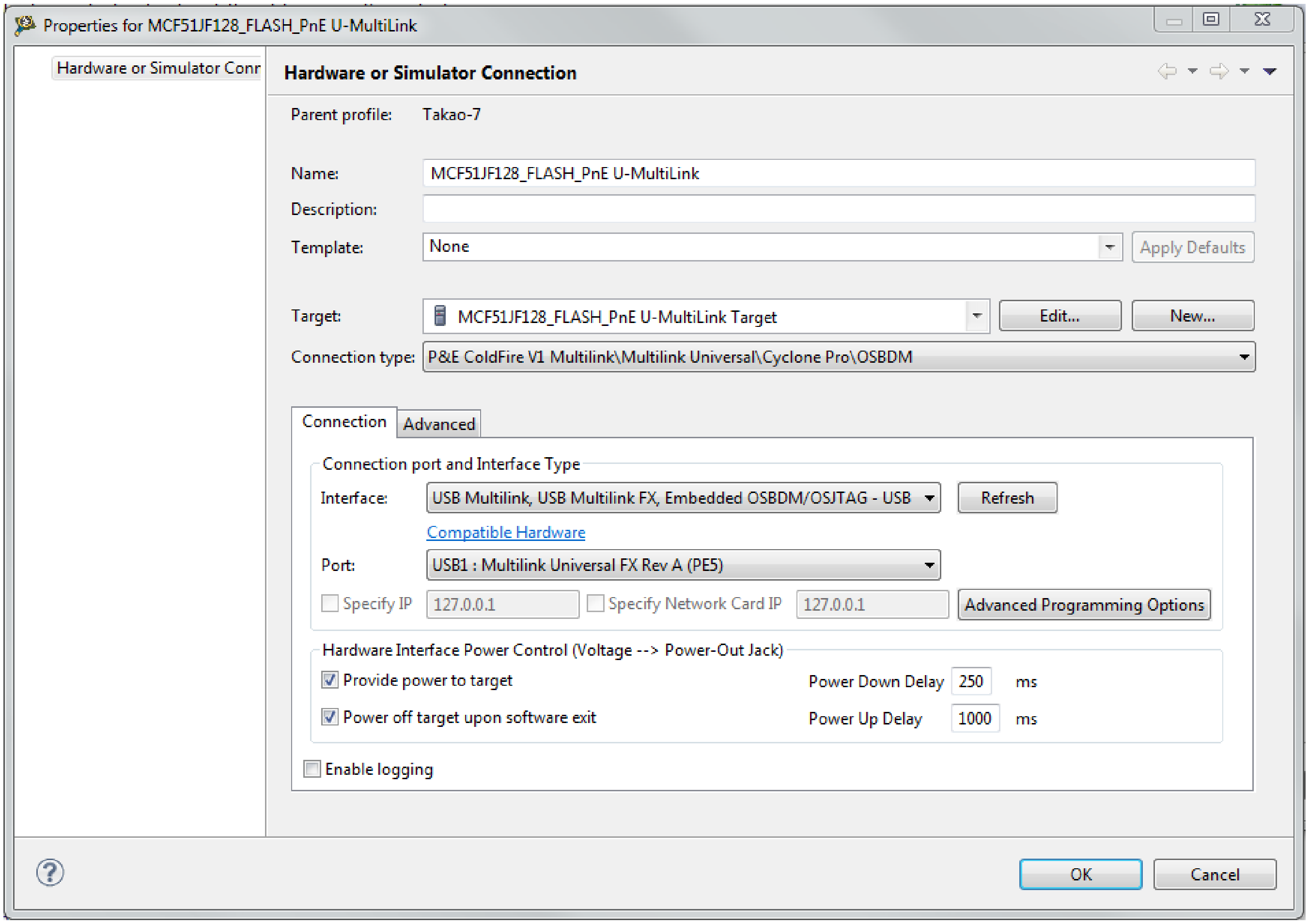
11.1.3.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

11.1.3.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V1 Multilink/Multilink Universal/Cyclone Pro/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF51JF128_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1

□ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

Power Down Delay 250 ms

☑ Power off target upon software exit

Power Up Delay 1000 ms

□ Enable logging

Figure 11-5. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

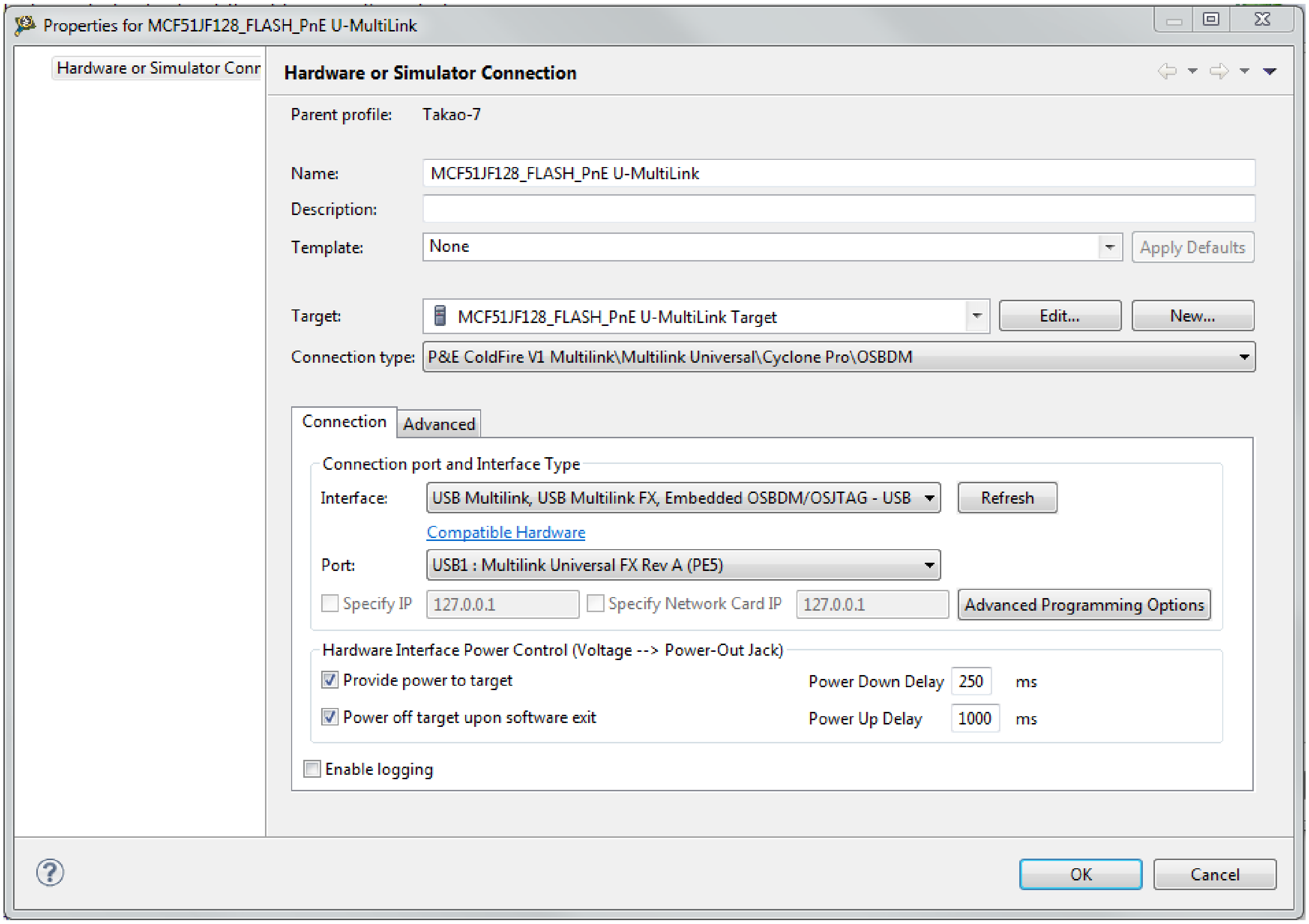
11.1.3.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

11.1.3.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V1 Multilink/Multilink Universal/Cyclone Pro/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF51JF128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

□ Enable logging

Figure 11-6. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone - Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

11.1.3.1.5 Open Source BDM

Freescale supplies certain development boards with an integrated debug circuit based on

Open Source BDM. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink. The development board also derives its power from the USB Bus.

The Open Source BDM circuit design (OSBDM-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

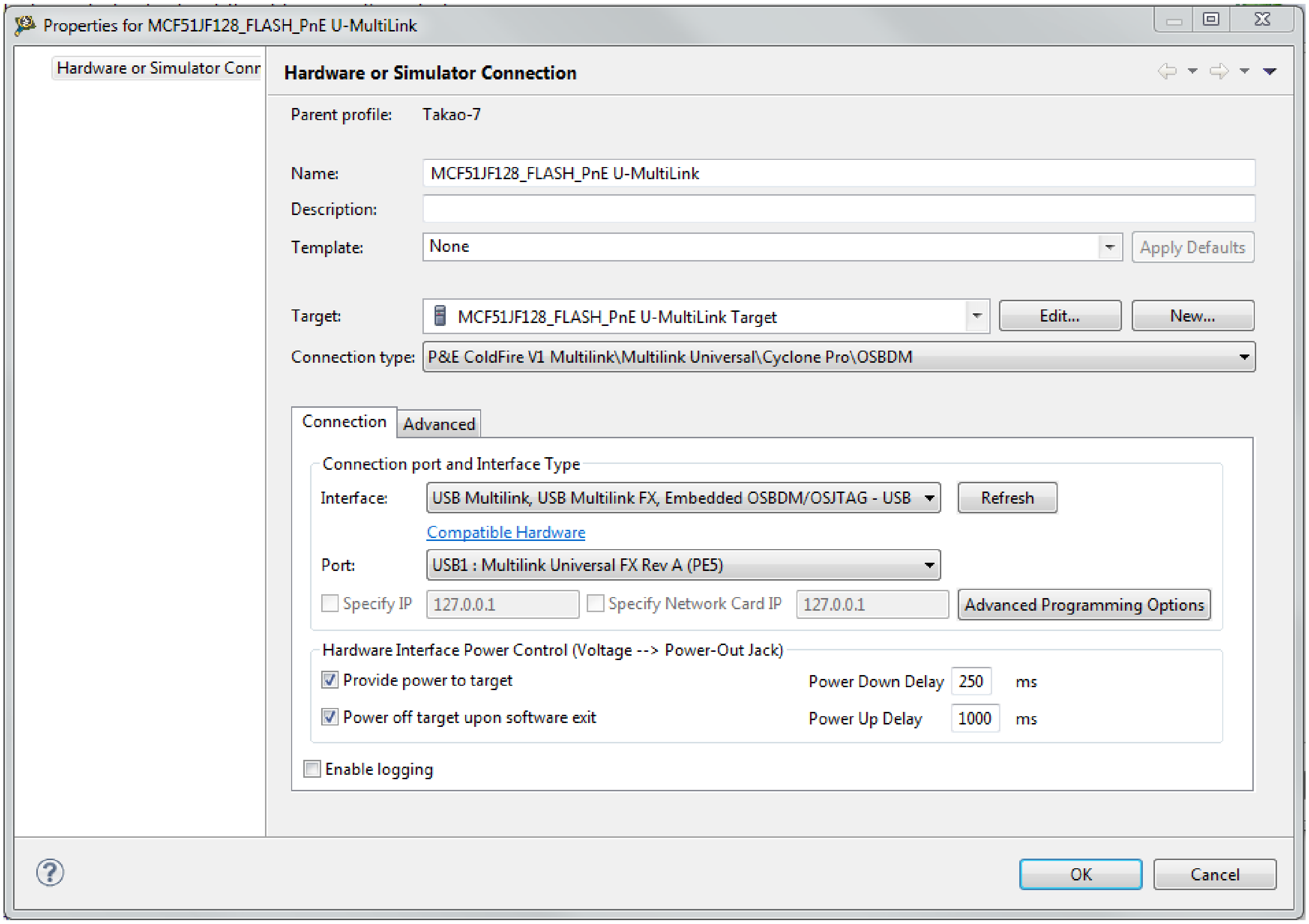
Integration with CodeWarrior is handled via the "Open Source BDM" connection. P&E has integrated the Open Source BDM support into the same connection that supports both the USB Multilink and the Cyclone. All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSBDM (albeit at a lower data rate).

The Open Source BDM Connection setting permits a connection to Open Source BDM devices. Open Source BDM mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

11.1.3.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V1 Multilink/Multilink Universal/Cyclone/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF51JF128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF51JF128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF51JF128_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V1 Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

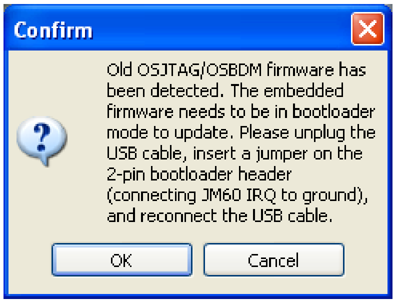
□ Enable logging

Figure 11-7. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

11.1.3.1.5.2 OSBDM Firmware Update

All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSBDM hardware interfaces. Whenever an OSBDM-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSBDM firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.

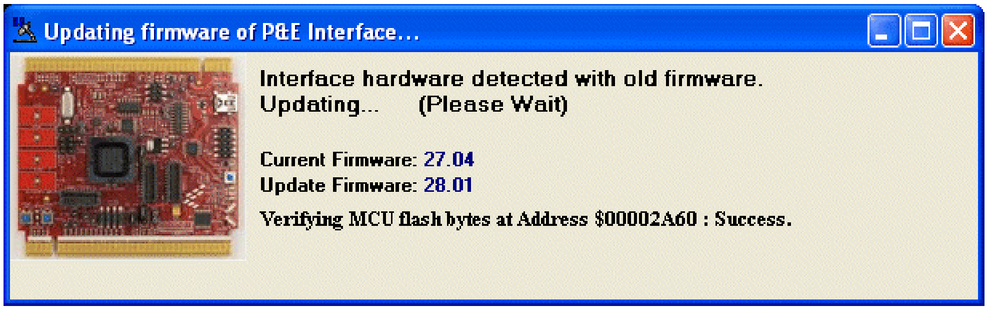


Confirm

Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 11-8. Old OSBDM Firmware Detected

To update the firmware, the OSBDM device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSBDM-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSBDM device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSBDM device and click on the OK button. If done correctly, the automatic firmware update will occur.



Updating firmware of P&E Interface...

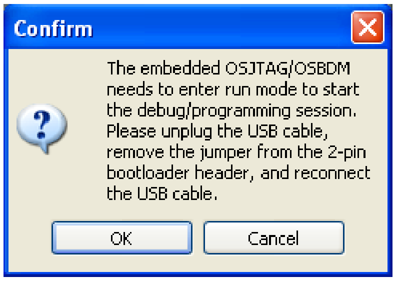
Interface hardware detected with old firmware.
Updating... (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01

Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 11-9. OSBDM Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSBDM device must exit Bootloader mode and enter into Run mode.



Confirm

The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

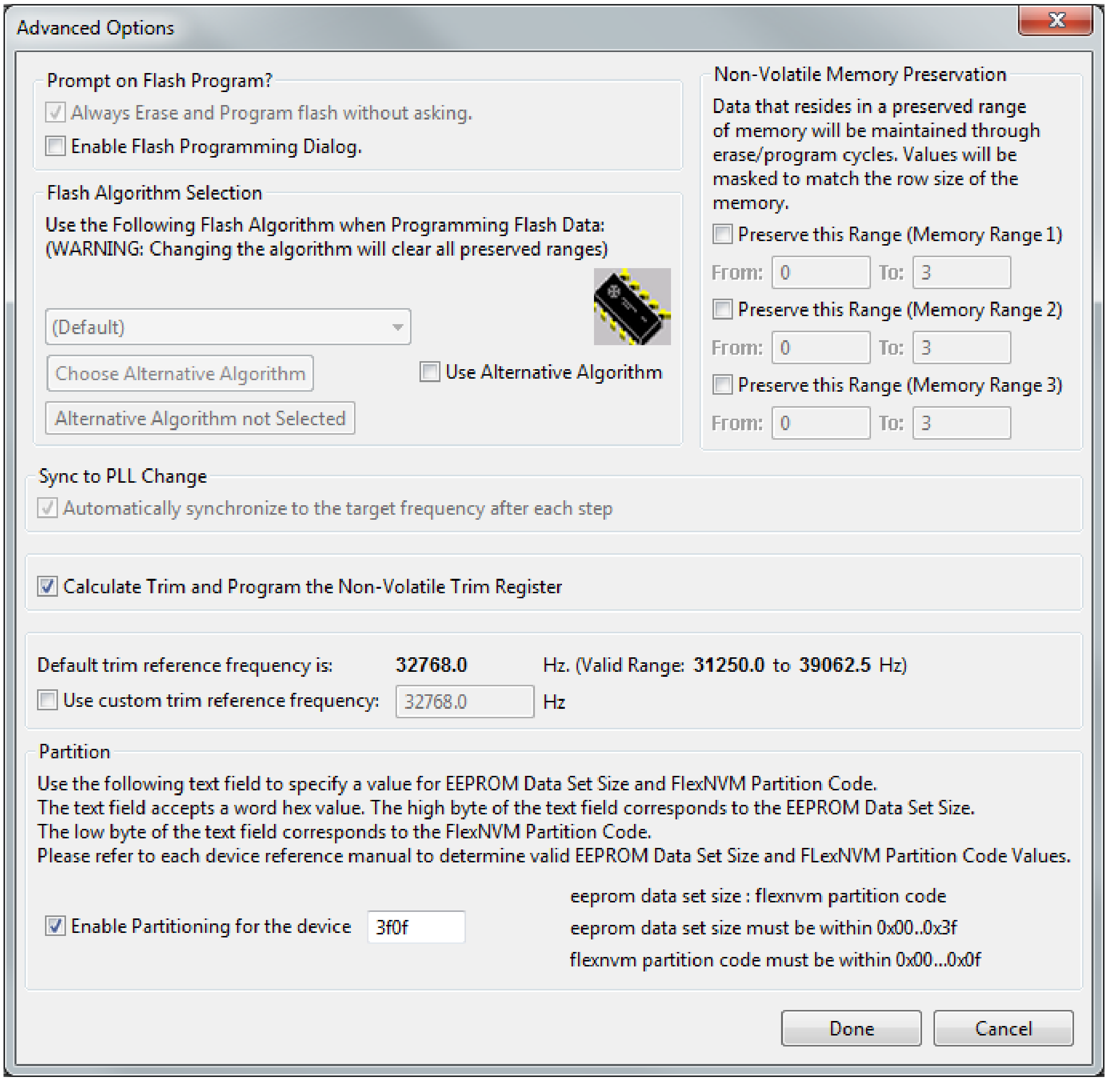
Figure 11-10. Start OSBDM Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSBDM device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSBDM firmware. If for any reason you experience difficulty performing OSBDM firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

11.1.3.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the Flash programming procedure.



Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

□ Use Alternative Algorithm

Alternative Algorithm not Selected

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 0 To: 3

□ Preserve this Range (Memory Range 2)

From: 0 To: 3

□ Preserve this Range (Memory Range 3)

From: 0 To: 3

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

☑ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: **32768.0** Hz. (Valid Range: **31250.0** to **39062.5** Hz)

□ Use custom trim reference frequency: 32768.0 Hz

Partition

Use the following text field to specify a value for EEPROM Data Set Size and FlexNVM Partition Code.
The text field accepts a word hex value. The high byte of the text field corresponds to the EEPROM Data Set Size.
The low byte of the text field corresponds to the FlexNVM Partition Code.
Please refer to each device reference manual to determine valid EEPROM Data Set Size and FLexNVM Partition Code Values.

☑ Enable Partitioning for the device 3f0f

eeprom data set size : flexnvm partition code
eeprom data set size must be within 0x00..0x3f
flexnvm partition code must be within 0x00...0x0f

Figure 11-11. Advanced Options Dialog Box

11.1.3.2.1 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

11.1.3.2.2 Trim Options

The Calculate Trim and Program the Non-Volatile Trim Register checkbox enables automatic calculation and programming of the trim value to a designated Non-Volatile memory location.

11.1.3.2.3 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the entire EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase and re-programmed immediately afterwards, thereby preserving the data in these ranges. Any attempts to program data into a preserved range are ignored. When entering an address into the preserved range field (hexadecimal input is required), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

11.1.3.2.4 Sync to PLL Change Checkbox

The debugger requires the Sync to PLL Change to synchronize the software/hardware connection with the microprocessor during the Flash erase/program procedure.

11.1.3.2.5 Calculate and Program Non-Volatile Trim

The checkbox gives you the option of trimming device to default center frequency. If this checkbox is selected, a calculated trim frequency will be programmed to a dedicated non-volatile memory location during the next debugging session.

11.1.3.2.6 Custom Trim

When the checkbox is checked, you have the ability to input a custom center frequency within an allowed range for a given device. A trim value based on this frequency will be calculated and programming into dedicated non-volatile memory location during the next debug session.

11.1.3.2.7 Alternative Algorithm Functionality

Once you create a project for a specific HCS08/RS08/CFV1 microprocessor, the debugger specifies a default algorithm to use during all Flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the `<CW_Install>/MCU/bin/plugins/support/Coldfire/gdi/P&E` directory.

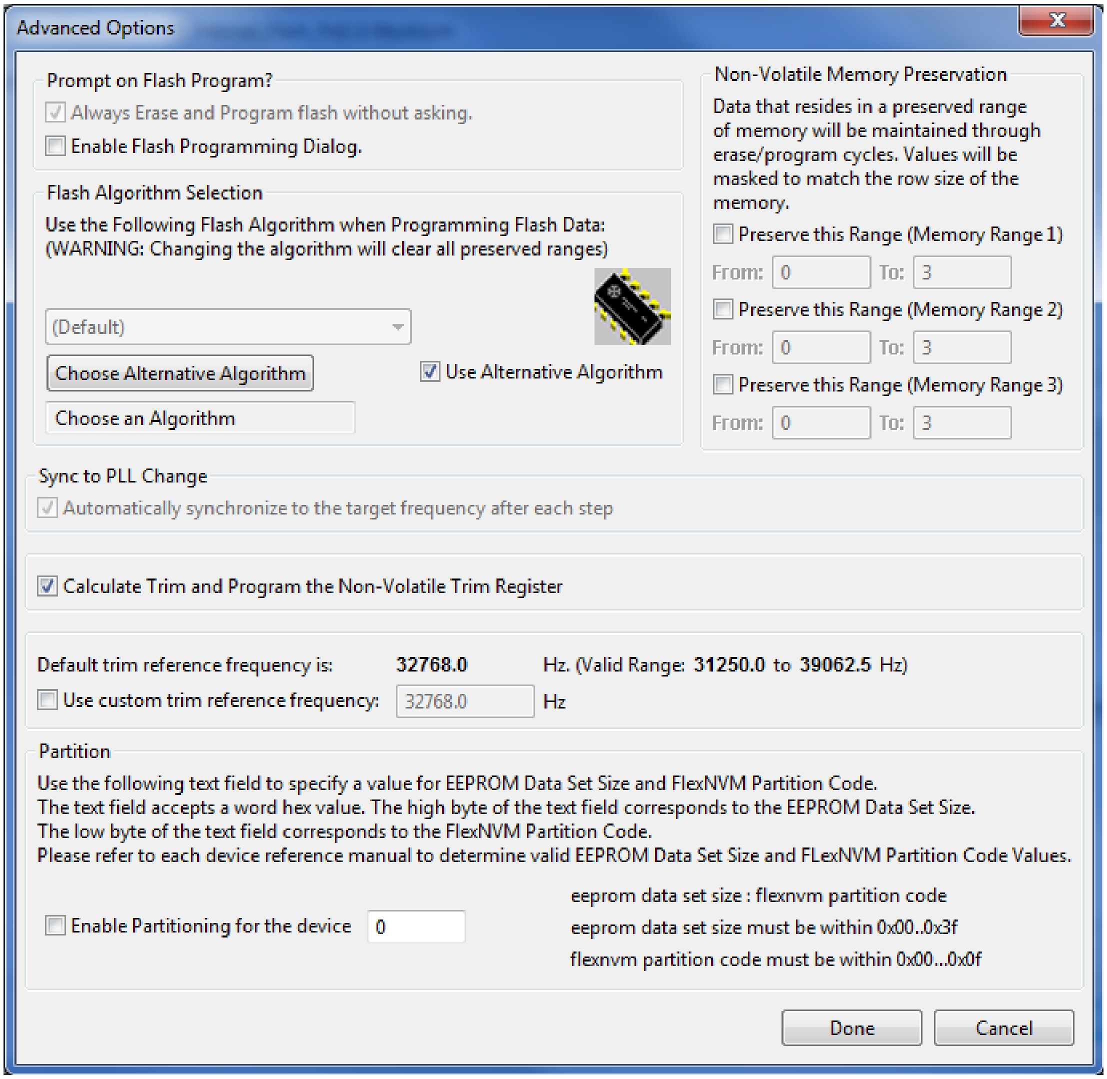
However, the default algorithm may be overridden via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. You can use this feature to select a custom programming algorithm, or simply select another one of P&E's many programming algorithms for use with a specific project.

CAUTION

> Selecting the wrong programming algorithm may damage their device, lead to under/over programming situations, or simply not program portions of the project file. It is therefore recommended using the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. Check the Use Alternative Algorithm checkbox.



Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

☑ Use Alternative Algorithm

Choose an Algorithm

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)
From: 0 To: 3

□ Preserve this Range (Memory Range 2)
From: 0 To: 3

□ Preserve this Range (Memory Range 3)
From: 0 To: 3

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

☑ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: **32768.0** Hz. (Valid Range: **31250.0** to **39062.5** Hz)

□ Use custom trim reference frequency: 32768.0 Hz

Partition

Use the following text field to specify a value for EEPROM Data Set Size and FlexNVM Partition Code.
The text field accepts a word hex value. The high byte of the text field corresponds to the EEPROM Data Set Size.
The low byte of the text field corresponds to the FlexNVM Partition Code.
Please refer to each device reference manual to determine valid EEPROM Data Set Size and FLexNVM Partition Code Values.

□ Enable Partitioning for the device 0

eeprom data set size : flexnvm partition code
eeprom data set size must be within 0x00..0x3f
flexnvm partition code must be within 0x00...0x0f

Figure 11-12. Advanced Options - Alternative Algorithm Checkbox

2. Click the **Choose Alternative Algorithm** button, which lets you browse for an alternative algorithm.
3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the **Choose Alternative Algorithm** button.
4. At this point, the current project performs all future Flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

11.1.3.3 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug but CodeWarrior cannot connect to the interface hardware specified in the Launch Configuration dialog box. To select the P&E Multilink/Cyclone /OSBDM as your debugger connection::

1. Select the P&E device that you are using from the first drop-down menu and click **Refresh**.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the Cyclone Power Control panel to configure the power and delay settings (Cyclone only).
4. Click the Retry button.

11.1.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.

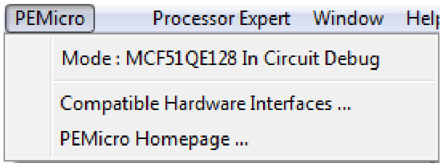

PEMicro
Processor Expert
Window
Help
Mode : MCF51QE128 In Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 11-13. Additional Connection Menu Options

Chapter 12
Connections - ColdFire V2/3/4

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the ColdFire V2/3/4 target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with the ColdFire V2/3/4 device family.

The topics in this chapter are:

12.1 P&E Hardware Interface Connection for ColdFire V234

This section describes the CFV234 P&E Connection options. The P&E Connection setting permits a connection to CFV234 Freescale devices via P&E Multilink, Cyclone , Tracelink, and OSBDM hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

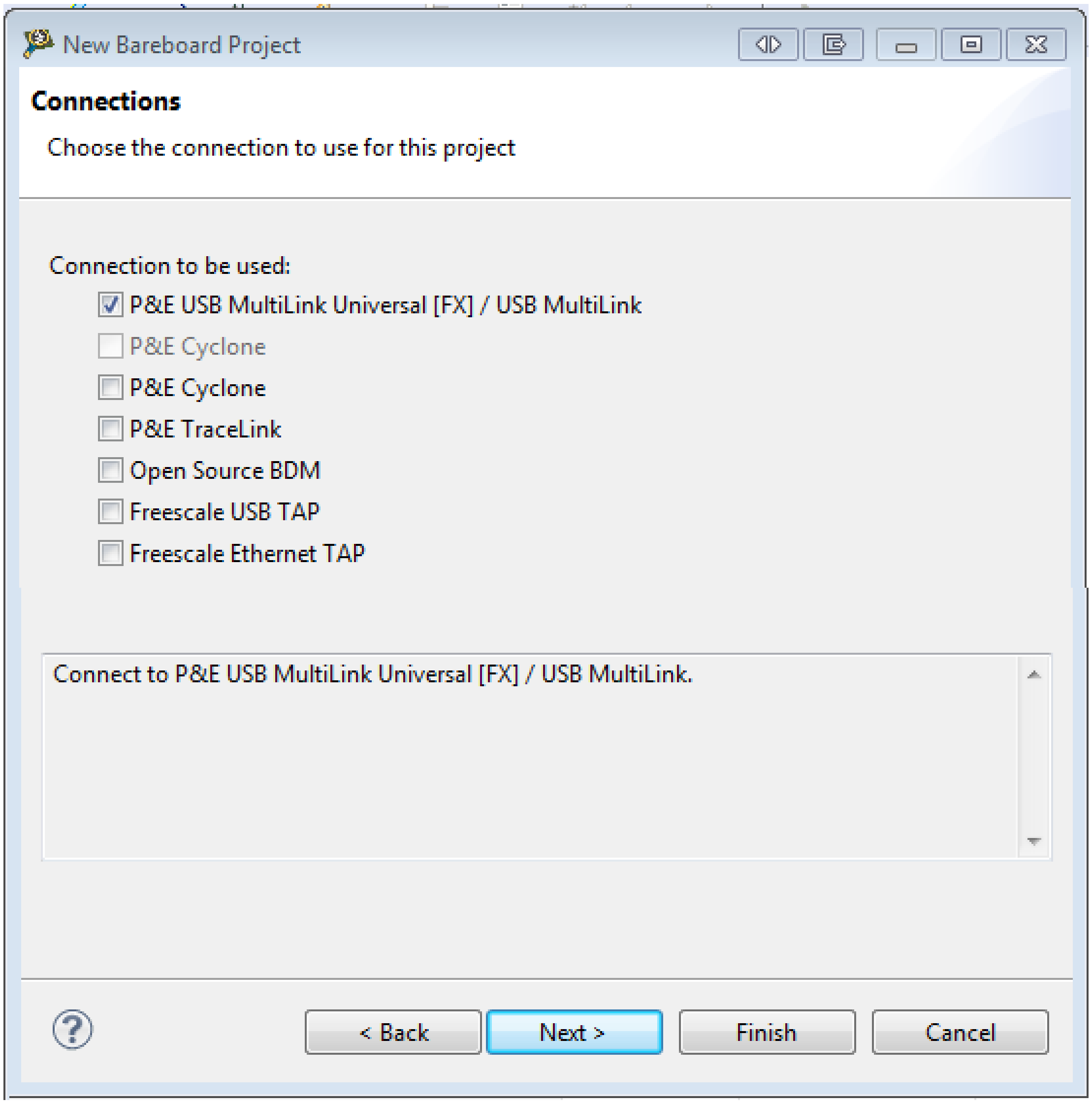
This topic describes all P&E Multilink/Cyclone /Tracelink/OSBDM connection options, which are common to all P&E USB Multilink Universal [FX]/ USB Multilink, P&E Cyclone Serial, P&E Cyclone USB, P&E Cyclone Ethernet, P&E TraceLink USB, P&E TraceLink Ethernet, and Open Source BDM

The options include:

- New Project Wizard
- Launch Configuration Settings
- Changing P&E Connection Settings

12.1.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.



New Bareboard Project

Connections

Choose the connection to use for this project

Connection to be used:

☑ P&E USB MultiLink Universal [FX] / USB MultiLink
□ P&E Cyclone
□ P&E Cyclone
□ P&E TraceLink
□ Open Source BDM
□ Freescale USB TAP
□ Freescale Ethernet TAP

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 12-1. New project wizard

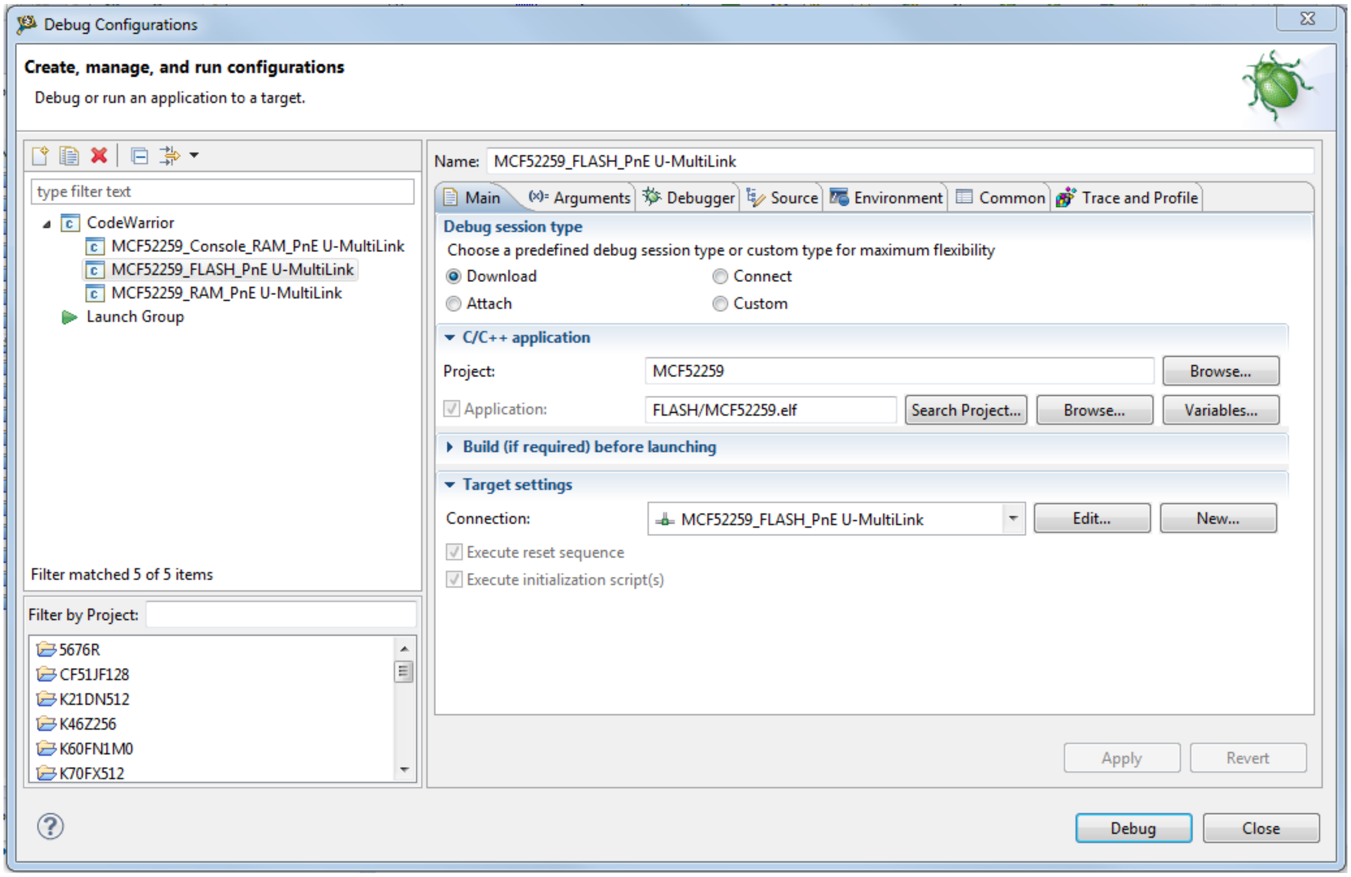
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

12.1.2 Launch Configuration Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - MCF52259_Console_RAM_PnE U-MultiLink
  - MCF52259_FLASH_PnE U-MultiLink
  - MCF52259_RAM_PnE U-MultiLink
- Launch Group

Filter matched 5 of 5 items

Filter by Project:

- 5676R
- CF51JF128
- K21DN512
- K46Z256
- K60FN1M0
- K70FX512

Name: MCF52259_FLASH_PnE U-MultiLink

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: MCF52259

☑ Application: FLASH/MCF52259.elf

Build (if required) before launching

Target settings

Connection: MCF52259_FLASH_PnE U-MultiLink

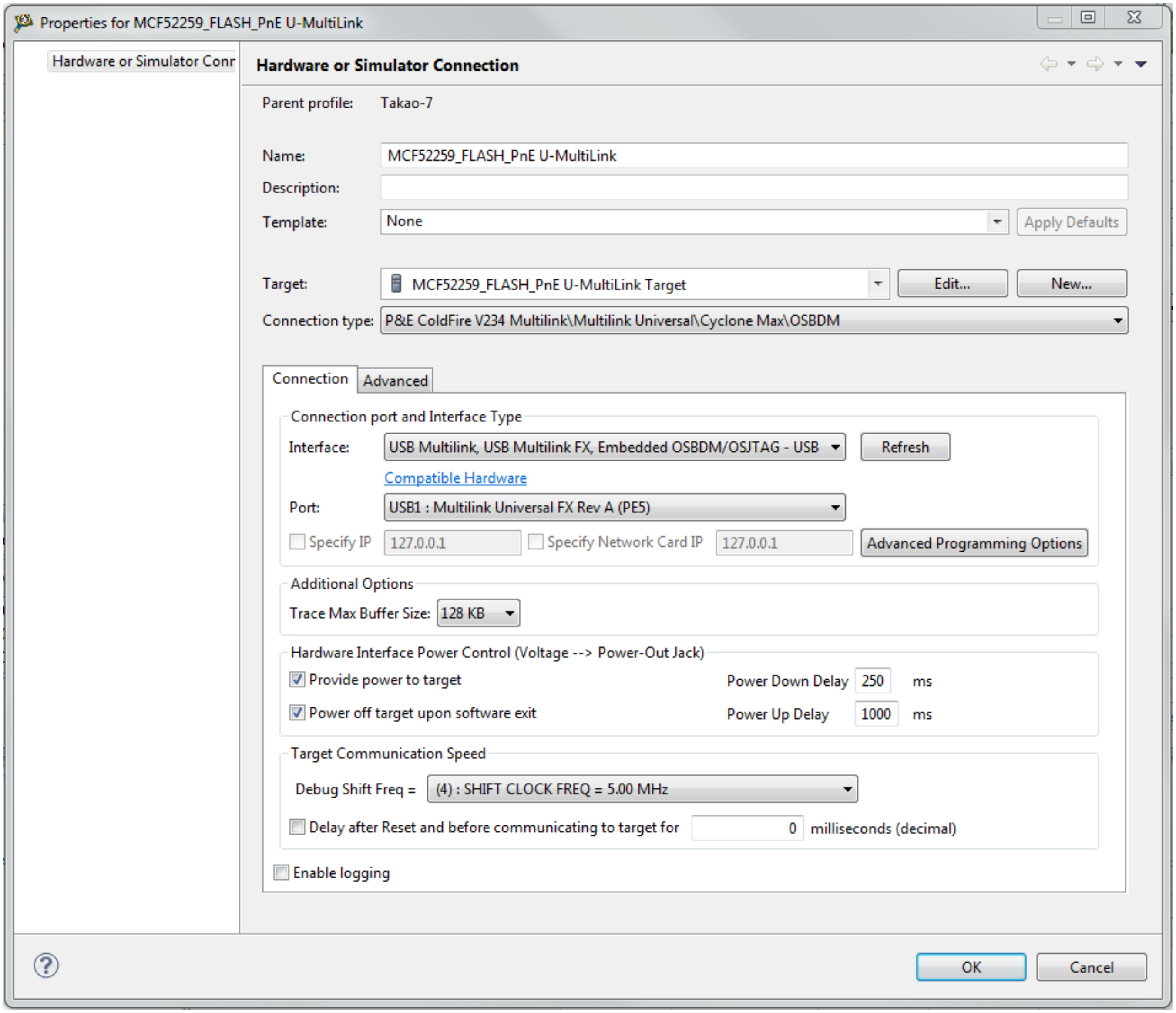
☑ Execute reset sequence

☑ Execute initialization script(s)

Figure 12-2. Debug Configuration Dialog Box

2. In the left column, select the project type you would like to set the launch configurations.

3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear.
5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

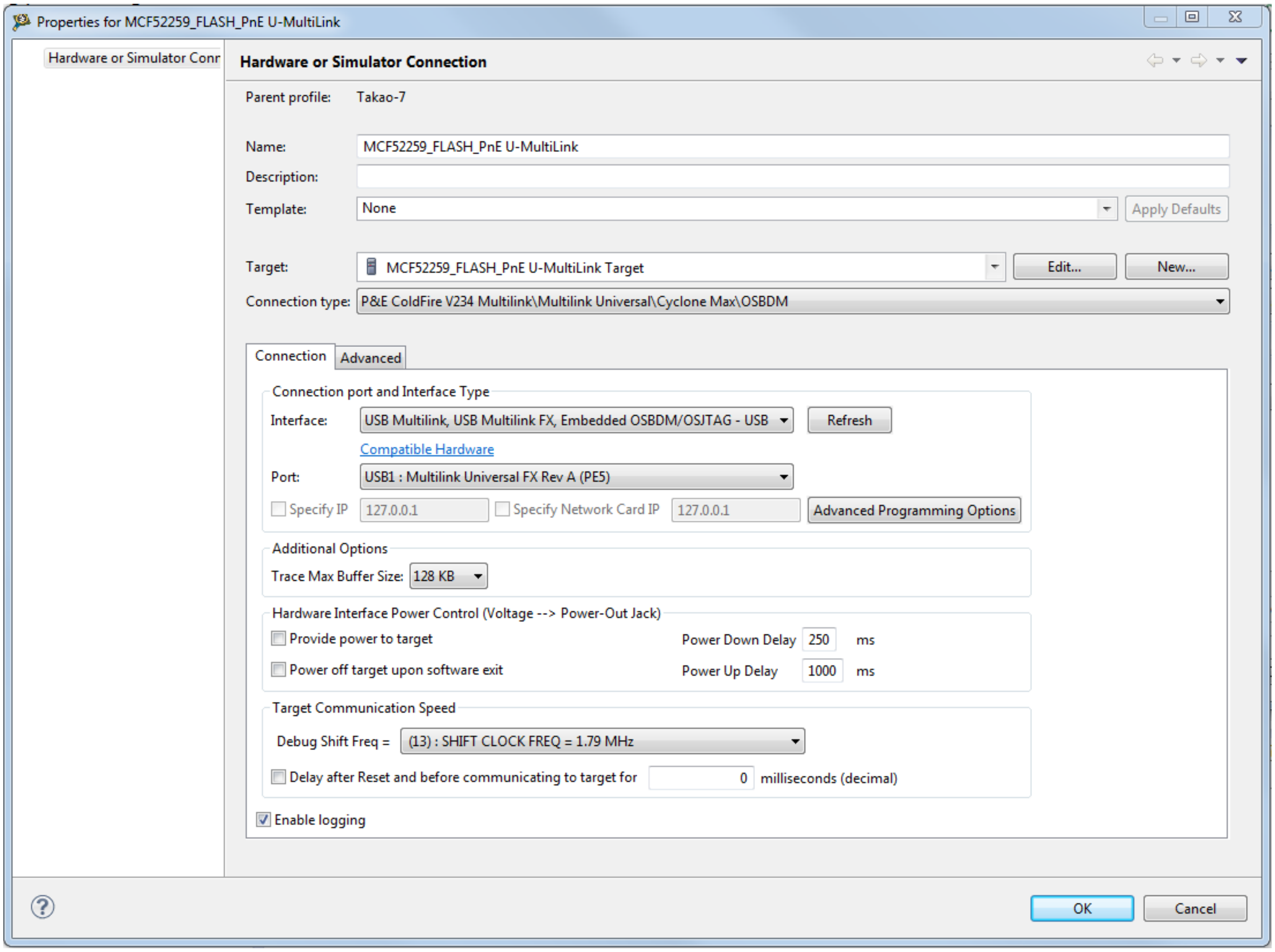
□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 12-3. Launch Configuration Dialog Box

12.1.3 Changing P&E Connection Settings

All connection settings for P&E hardware interfaces are configured in the Launch Configurations dialog box.


Properties for MCF52259_FLASH_PnE U-MultiLink
Hardware or Simulator Conr
Apply Defaults
Edit...
New...
Refresh
Compatible Hardware
Advanced Programming Options
OK
Cancel

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

☐ Specify IP 127.0.0.1 ☐ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☐ Provide power to target Power Down Delay 250 ms

☐ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (13) : SHIFT CLOCK FREQ = 1.79 MHz

☐ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 12-4. P&E CFV234 Launch Configuration Dialog Box

The following table describes the options for this view.

Table 12-1. Connection Parameter Options for P&E Multilink/Cyclone /Tracelink/OSBDM

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB ColdFire Multilink - USB Port • USB Multilink Universal [FX]- USB Port NOTE: The USB Multilink Universal and USB Multilink Universal FX can conveniently support all Freescale architectures found in the current CodeWarrior 10 version • Cyclone - Serial Port • Cyclone - USB Port • Cyclone - Ethernet Port • TraceLink - USB Port

Table continues on the next page...

Table 12-1. Connection Parameter Options for P&E Multilink/Cyclone /Tracelink/OSBDM (continued)

Option	Description
	• TraceLink - Ethernet Port • OSBDM NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help
Specify IP (Cyclone and Tracelink Ethernet only)	Use this option to specify the IP address of a Cyclone or Tracelink outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port drop down box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone and Tracelink Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Provide power to target (Tracelink and USB Multilink Universal FX only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (Tracelink and USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Regulator Output Voltage (Tracelink only)	This option adjusts the output voltage that powers the hardware target. Select a voltage value from this option's list box. **CAUTION:** An improper voltage setting can damage the board.
Power down delay (Tracelink and USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (Tracelink and USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Debug Shift Freq.	This option lets you to set the debug shift clock speed of P&E's interfaces. This integer value may be used to determine the speed of communications according to the following equations: Cyclone / TraceLink : (50000000/(2*N +5)) Hz USB-ML-CF : (1000000/(N+1)) Hz USB Multilink Universal: (1000000/(N+1)) Hz USB ML Universal FX : (25000000/(N+1)) Hz OSBDM : Fixed Frequency The value n should be between 0 and 31. This shift clock takes effect after the commands in the top of the programming algorithm are

Table continues on the next page...

Table 12-1. Connection Parameter Options for P&E Multilink/Cyclone /Tracelink/OSBDM (continued)

Option	Description
	executed so that these commands can increase the target frequency and allow a faster shift clock. This clock can't generally exceed a div 4 of the processor bus frequency.
Delay After Reset	Specifies a delay after the programmer resets the target that we check to see if the part has properly gone into background debug mode. This is useful if the target has a reset driver which hold the MCU in reset after the programmer releases the reset line. The n value is a delay in milliseconds.
Trace Max Buffer Size	Only applicable for the P&E TraceLink interface. Configures the trace buffer capacity of the TraceLink. Smaller sizes result in faster trace upload times, whereas larger sizes allow the user to view more trace information.

To change P&E Connections settings, perform these steps:

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

NOTE

It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

   The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection of which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E ColdFire V234 Multilink/ Multilink Universal/Cyclone /OSBDM. The P&E connections settings will appear below.
8. Click Refresh to scan valid interface and port.

Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

NOTE

> The port displayed may vary depending on the interface. For example, if you select interface as Cyclone - Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Hardware Interface Power Control (Voltage --> Power -Out Jack) group.

NOTE

> This group will be enabled for the Tracelink and USB Multilink Universal FX interfaces only. For USB Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

- Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
- Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.
- Select a voltage value from the Regulator Output Voltage drop-down list. This adjusts the output voltage that powers the hardware target.

CAUTION

> An improper voltage setting can damage the board.

- Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
- Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
- Select a BDM speed using the drop-down box in the BDM Debug Shift Frequency.

NOTE

If you select a fast BDM speed, there may be scenarios where you may have difficulty communicating with your target device. You may want to experimentally select a BDM speed that is fast and yet have a good communication with your target.

- Click on the Delay after Reset checkbox and enter the desired delay (in milliseconds) in the text box.

  This option specifies the time interval to wait between resetting and communicating the target device.

- Click the OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.
- Click on the Close button to close the Debug Configuration dialog box.

12.1.3.1 P&E Hardware Interface Connection- Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]/ USB Multilink
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- P&E TraceLink USB
- P&E TraceLink Ethernet
- Open Source BDM

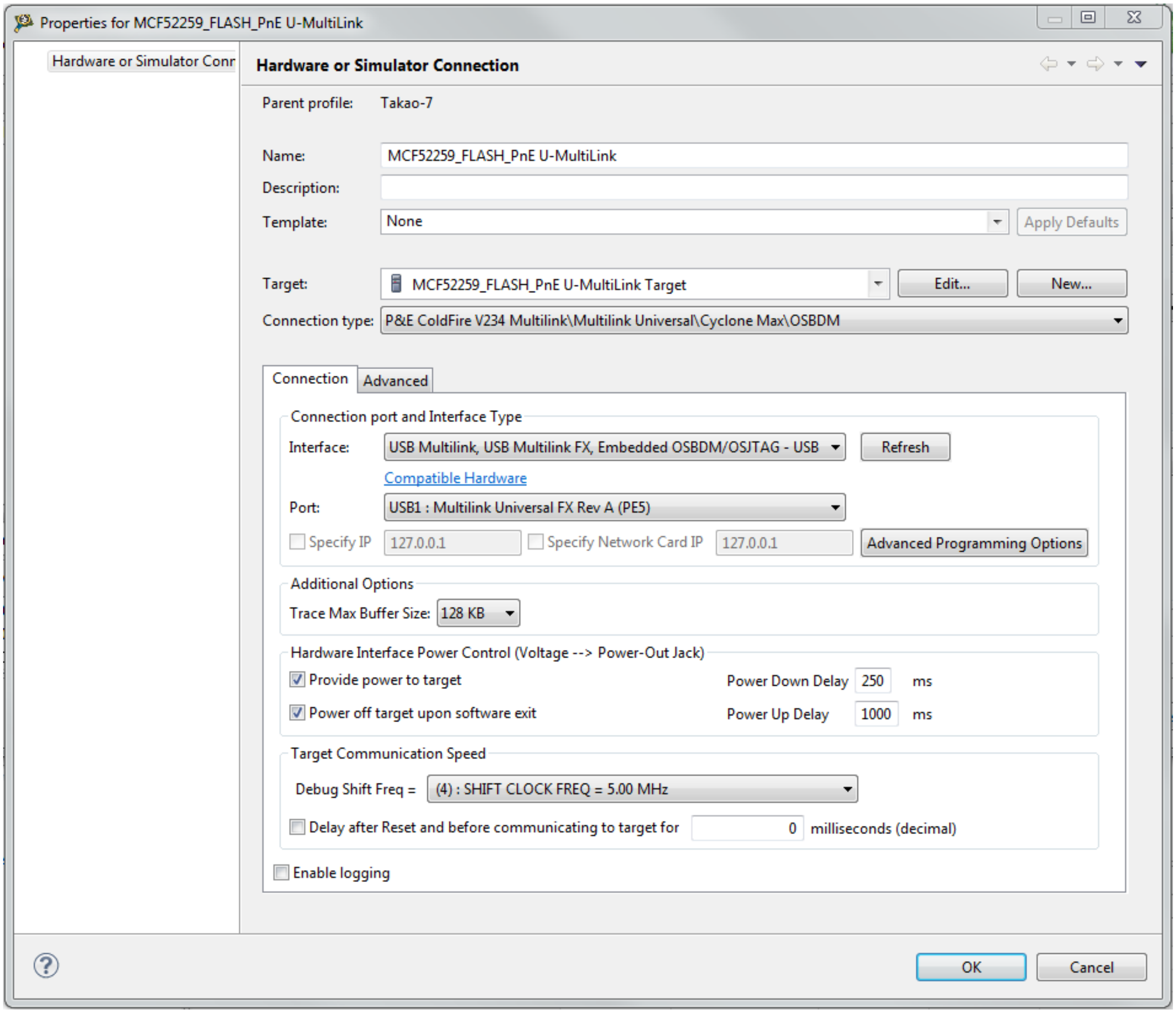
12.1.3.1.1 P&E USB Multilink Universal [FX]/ USB Multilink

The P&E USB Multilink Universal [FX]/ USB Multilink Connection setting permits a connection to USB Multilink devices, which include the P&E ColdFire Multilink, USB Multilink Universal, and the USB Multilink Universal FX. P&E USB Multilink Universal [FX]/ USB Multilink mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

12.1.3.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone /OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 12-5. P&E's Launch Configuration Dialog Box

To use P&E's USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

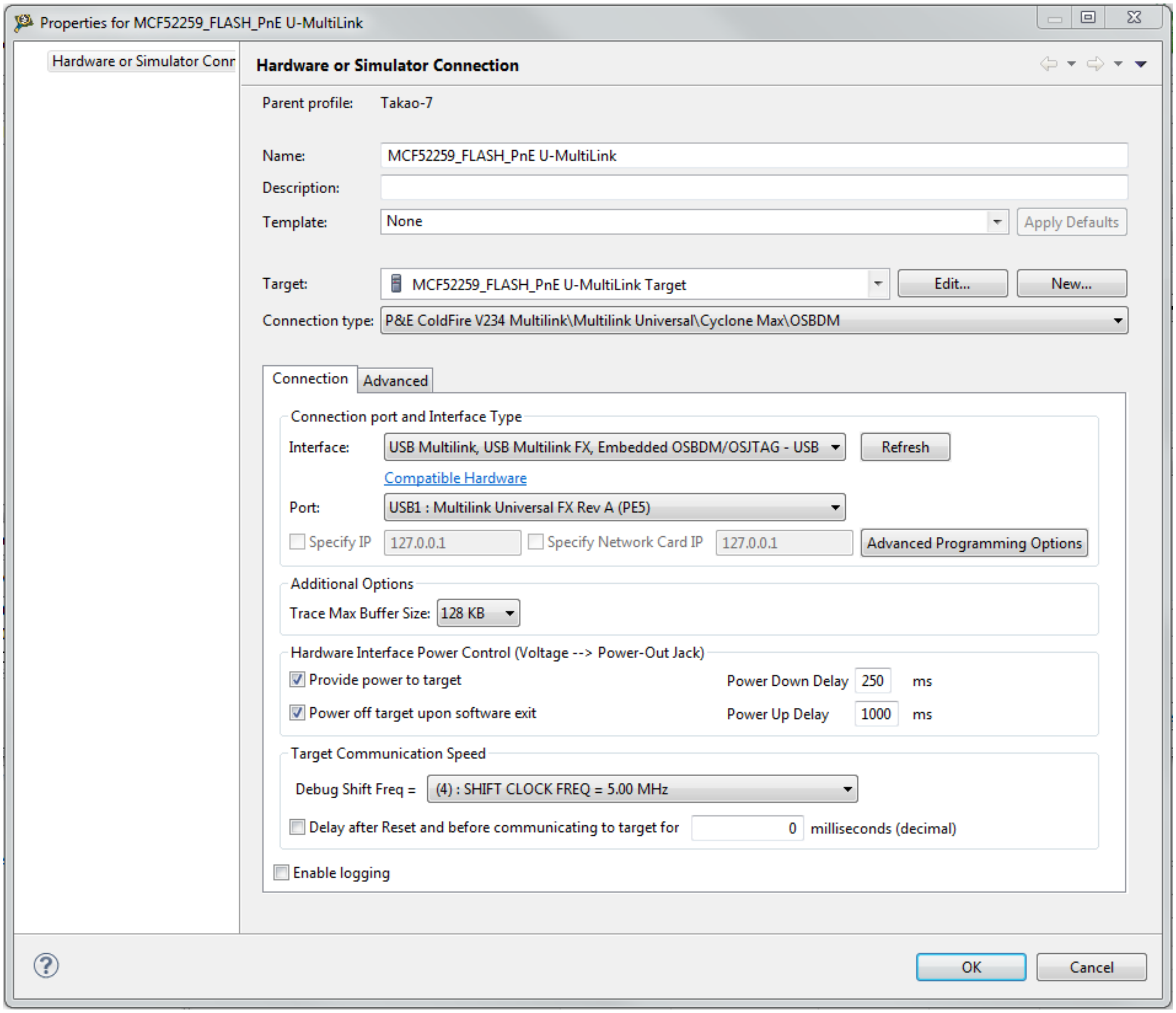
12.1.3.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

12.1.3.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 12-6. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone - Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

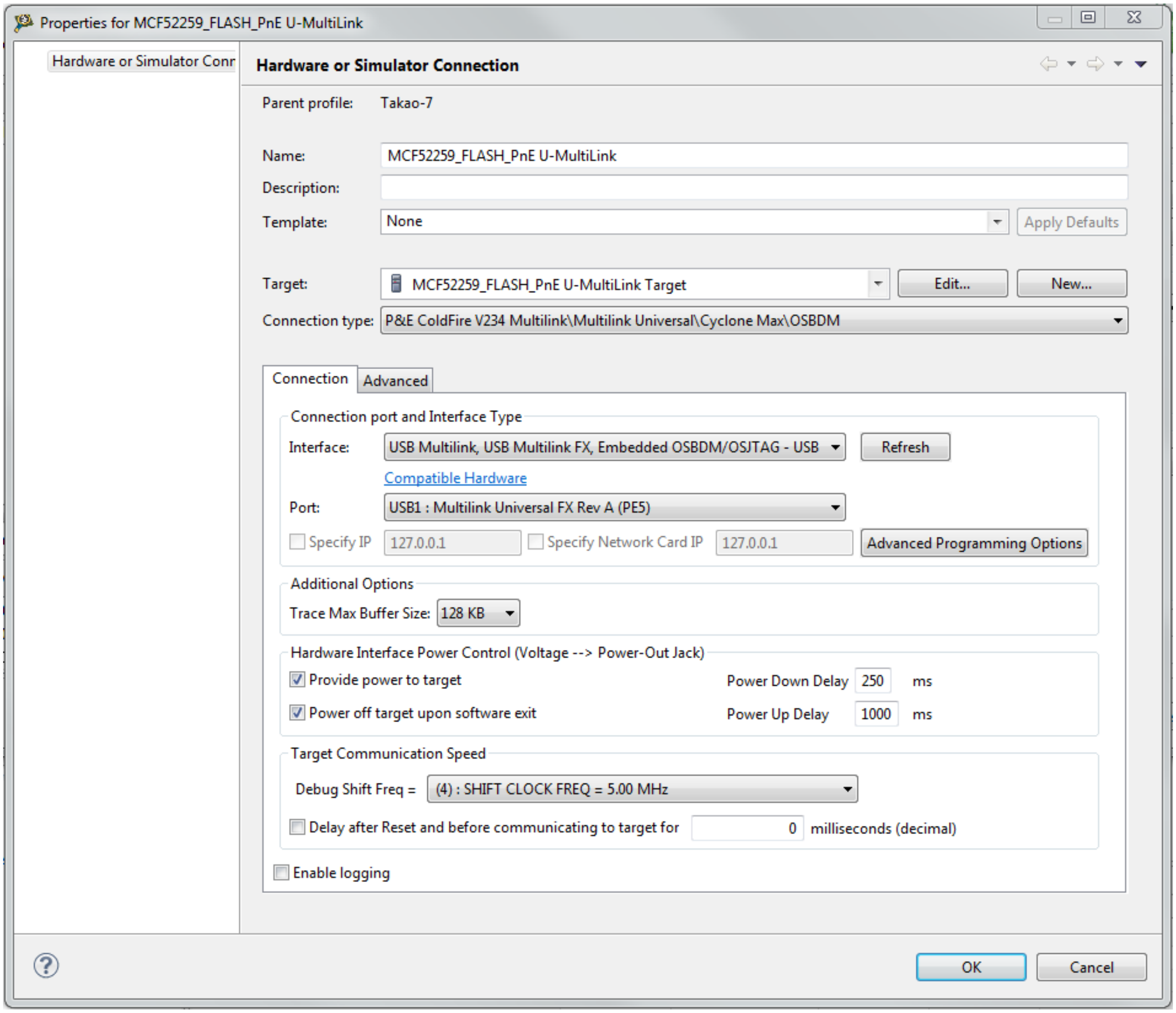
12.1.3.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

12.1.3.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None | Apply Defaults

Target: MCF52259_FLASH_PnE U-MultiLink Target | Edit... | New...

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB | Refresh

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1 | Advanced Programming Options

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

OK | Cancel

Figure 12-7. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

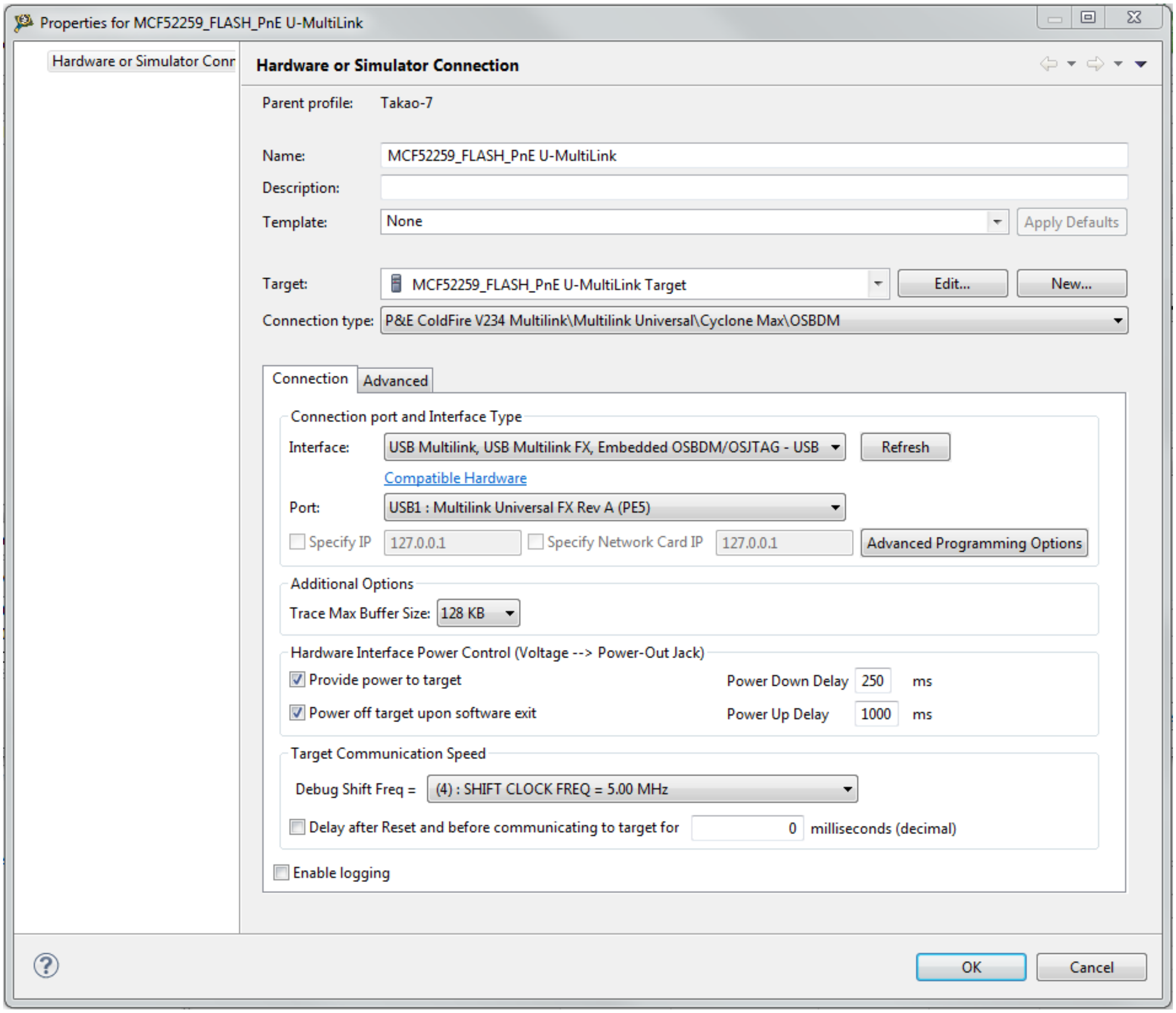
12.1.3.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

12.1.3.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.


Properties for MCF52259_FLASH_PnE U-MultiLink
Hardware or Simulator Conr
Apply Defaults
Edit...
New...
Connection
Advanced
Refresh
Compatible Hardware
Advanced Programming Options
OK
Cancel

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 12-8. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone - Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

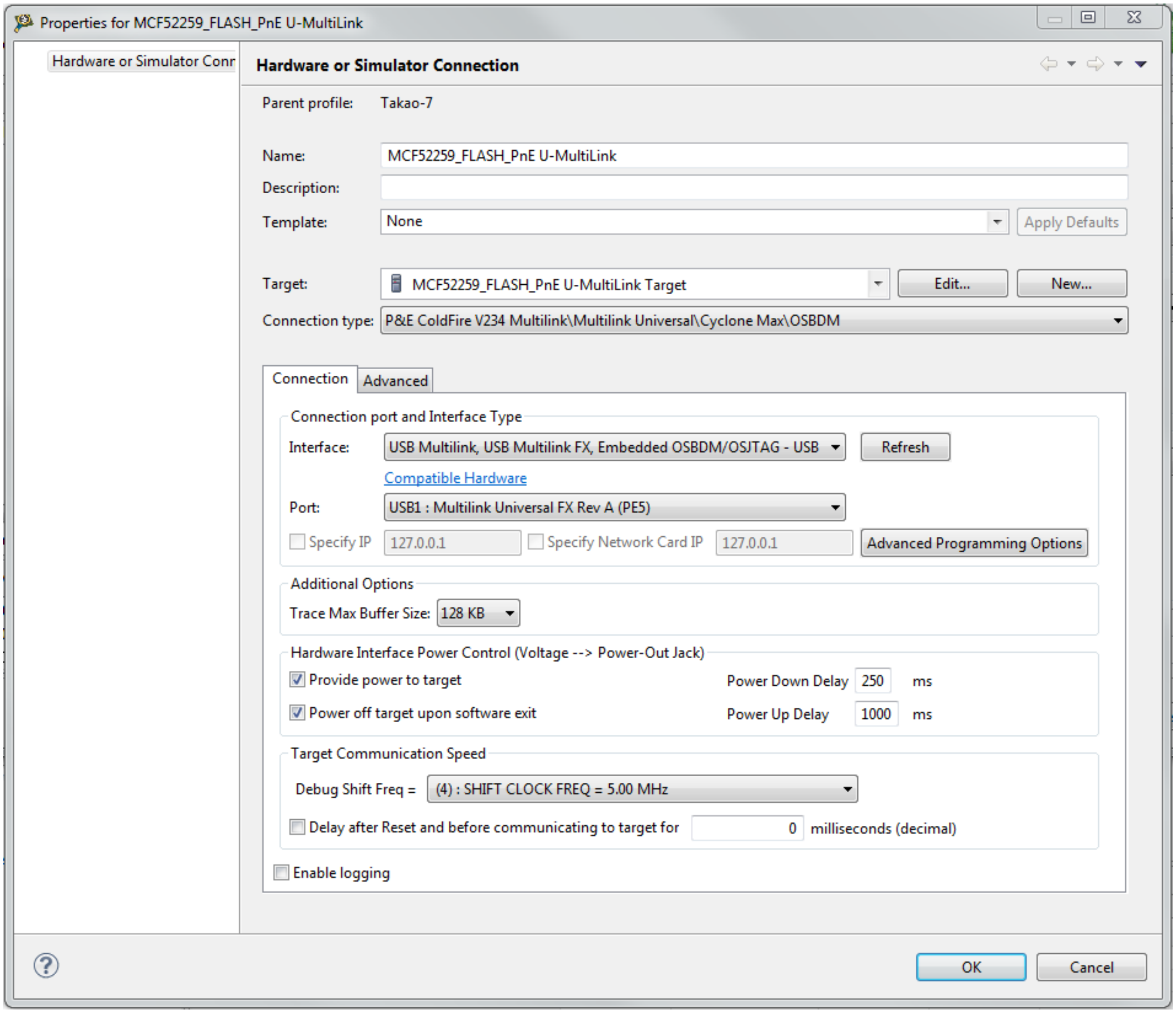
12.1.3.1.5 P&E TraceLink USB

The P&E TraceLink USB Connection setting permits a connection to TraceLink USB devices. P&E TraceLink USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. P&E's TraceLink can conveniently support all Freescale architectures found in the current CodeWarrior 10 version; however, it can only capture the external trace signals for the ColdFire V2/3/4 and Kinetis ARM architectures.

12.1.3.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 12-9. P&E's Launch Configuration Dialog Box

To use P&E's TraceLink USB, first connect your hardware interface to your computer, and then set the interface to TraceLink - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

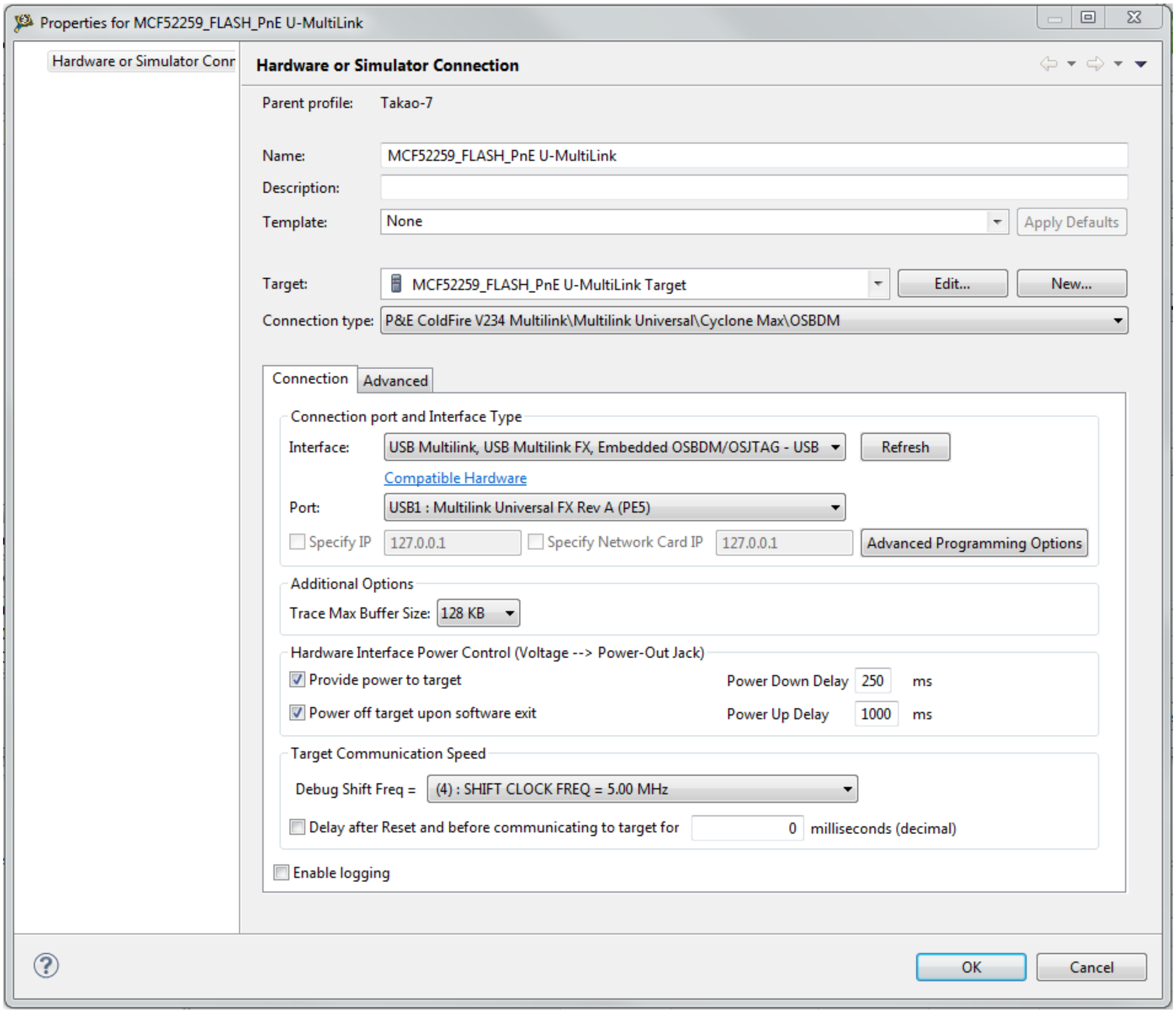
12.1.3.1.6 P&E TraceLink Ethernet

The P&E TraceLink Ethernet Connection setting permits a connection to TraceLink Ethernet devices. P&E TraceLink Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. P&E’s TraceLink can conveniently support all Freescale architectures found in the current CodeWarrior 10 version; however, it can only capture the external trace signals for the ColdFire V234 and Kinetis ARM architectures. For external trace captures, the P&E TraceLink must have PST_CLK and DDATA[3:0] pin connections from the ColdFire chip. Chips without these pins connected can still be debugged by the TraceLink within CodeWarrior.

12.1.3.1.6.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

☑ Power off target upon software exit

Power Down Delay 250 ms

Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 12-10. P&E's Launch Configuration Dialog Box

To use P&E's TraceLink Ethernet, first connect your hardware interface to your computer, and then set the interface to TraceLink - Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

12.1.3.1.7 Open Source BDM

Freescale supplies certain development boards with an integrated debug circuit based on Open Source BDM. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB ColdFire Multilink. The development board also derives its power from the USB Bus.

The Open Source BDM circuit design (OSBDM-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

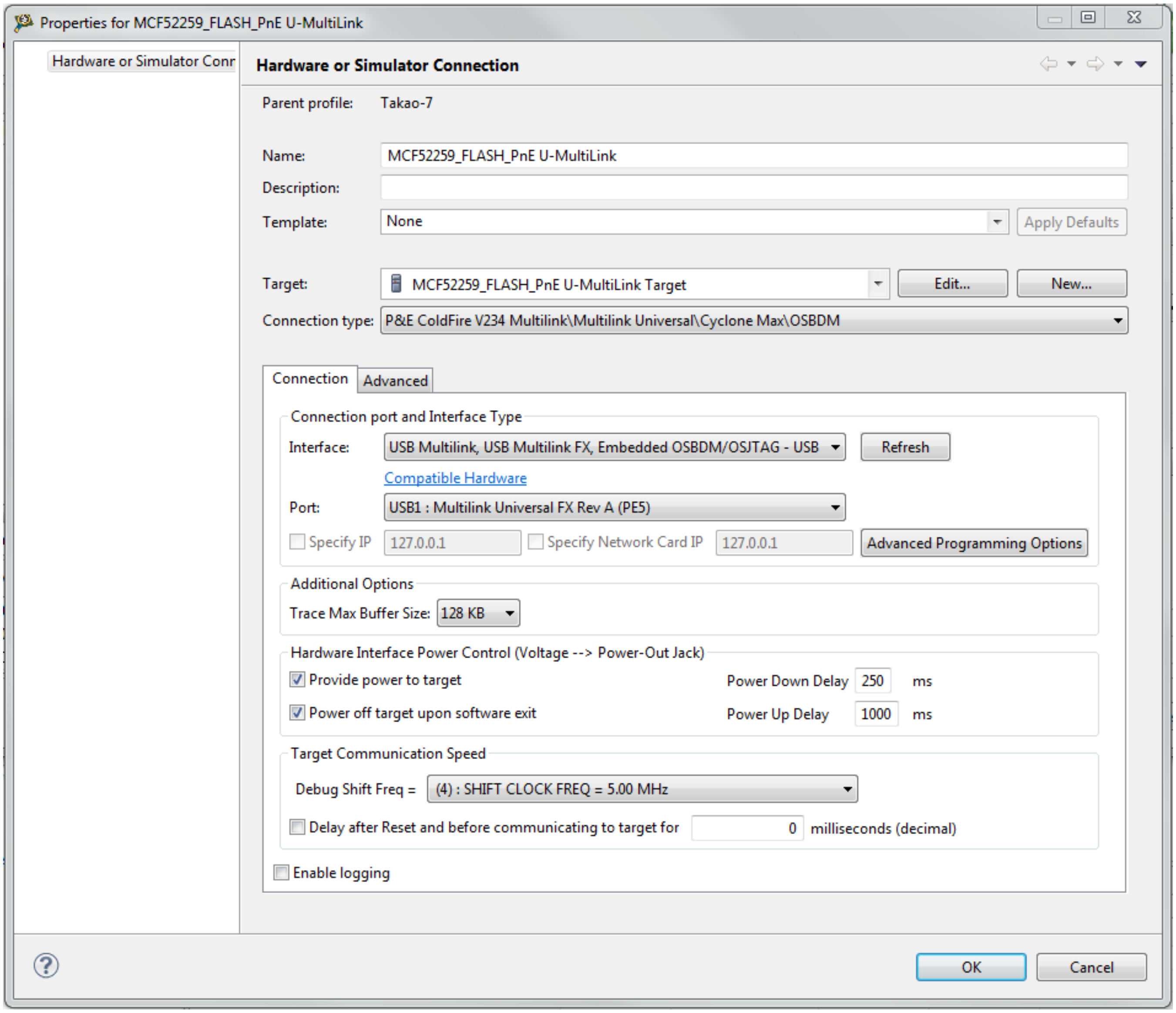
Integration with CodeWarrior is handled via the "Open Source BDM" connection. P&E has integrated the Open Source BDM support into the same connection that supports both the USB ColdFire Multilink and the Cyclone . All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSBDM (albeit at a lower data rate).

The Open Source BDM Connection setting permits a connection to Open Source BDM devices. Open Source BDM mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

12.1.3.1.7.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ColdFire V234 Multilink/Multilink Universal/ Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MCF52259_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF52259_FLASH_PnE U-MultiLink

Description:

Template: None

Target: MCF52259_FLASH_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

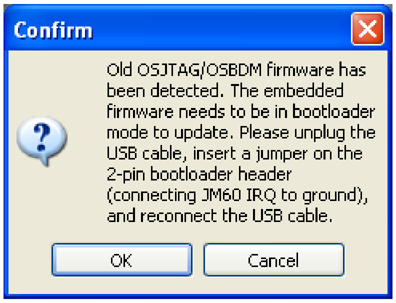
□ Enable logging

Figure 12-11. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

12.1.3.1.7.2 OSBDM Firmware Update

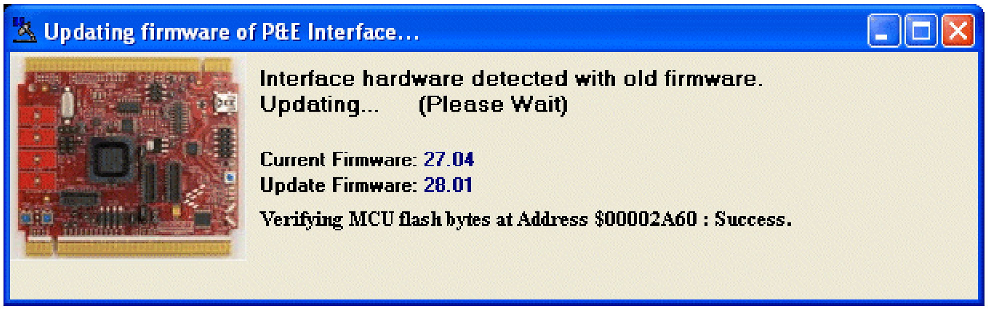
All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSBDM hardware interfaces. Whenever an OSBDM-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSBDM firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.



Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 12-12. Old OSBDM Firmware Detected

To update the firmware, the OSBDM device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSBDM-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSBDM device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSBDM device and click on the OK button. If done correctly, the automatic firmware update will occur.

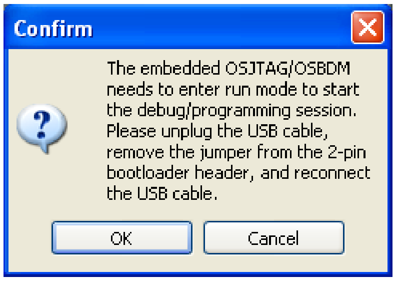


Interface hardware detected with old firmware.
Updating...   (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01
Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 12-13. OSBDM Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSBDM device must exit Bootloader mode and enter into Run mode.



The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

Figure 12-14. Start OSBDM Run Mode

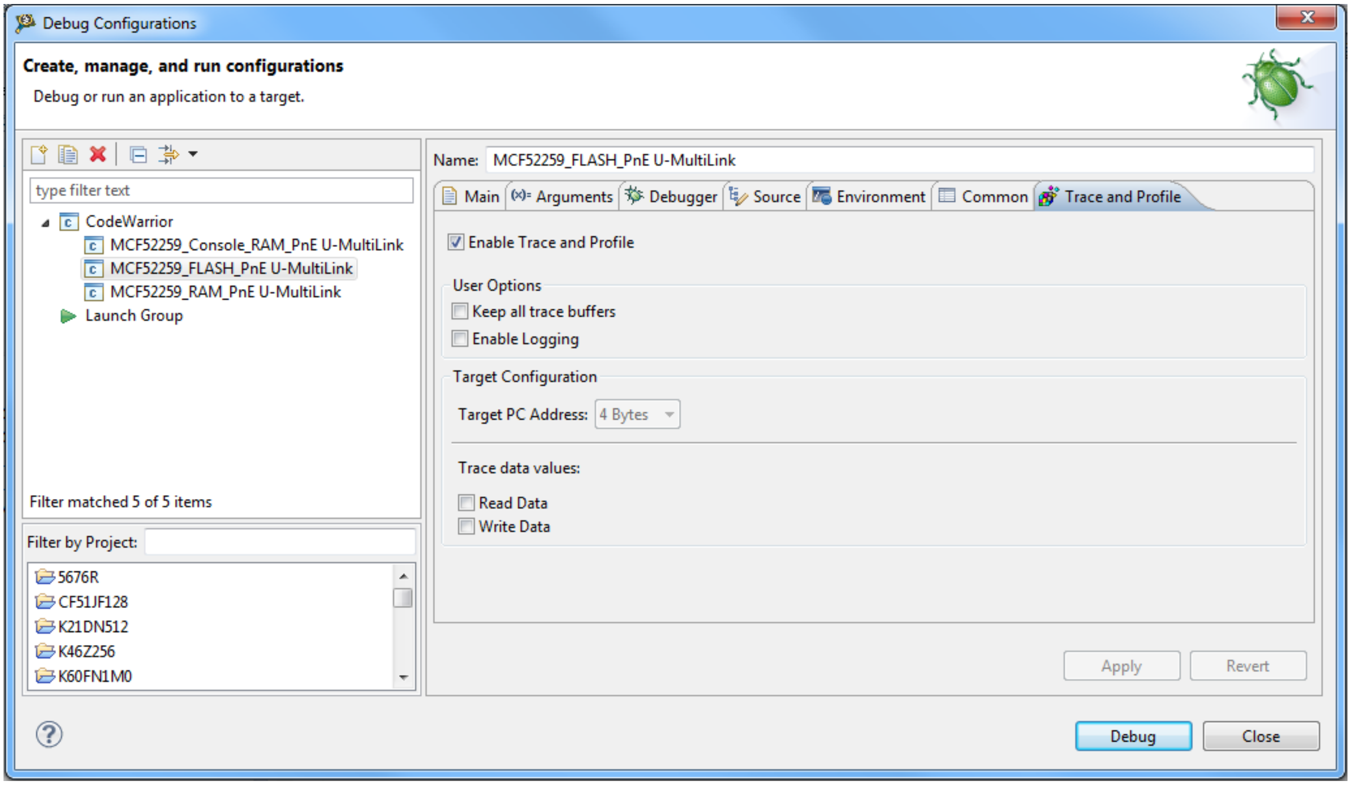
To enter Run Mode, the user must disconnect the USB cable from the OSBDM device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and Codewarrior will move onto programming or running the code.

The Codewarrior IDE layout will have the latest OSBDM firmware. If for any reason you experience difficulty performing OSBDM firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

12.1.3.1.8 Trace and Profile

The P&E TraceLink allows the user to capture real-time external trace information without having to stop or disturb the running application. This allows the user to see the real-time execution of their code by continuously recording the processor events. For external trace captures, the P&E TraceLink must have PST_CLK and DDATA[3:0] pin connections from the ColdFire chip. Chips without these pins connected can still be debugged by the TraceLink within CodeWarrior. The user must do the following to set up external trace captures using the TraceLink.

1. Right-click on the project and right-click -> Debug as -> Debug Configurations. The Debug Configuration dialog box will appear.
2. In the left column, select the project type for which you would like to set the TraceLink settings.
3. In the right column, select the Trace and Profile tab..



Create, manage, and run configurations

Debug or run an application to a target.

Name: MCF52259_FLASH_PnE U-MultiLink

☑ Enable Trace and Profile

User Options

☐ Keep all trace buffers

☐ Enable Logging

Target Configuration

Target PC Address: 4 Bytes

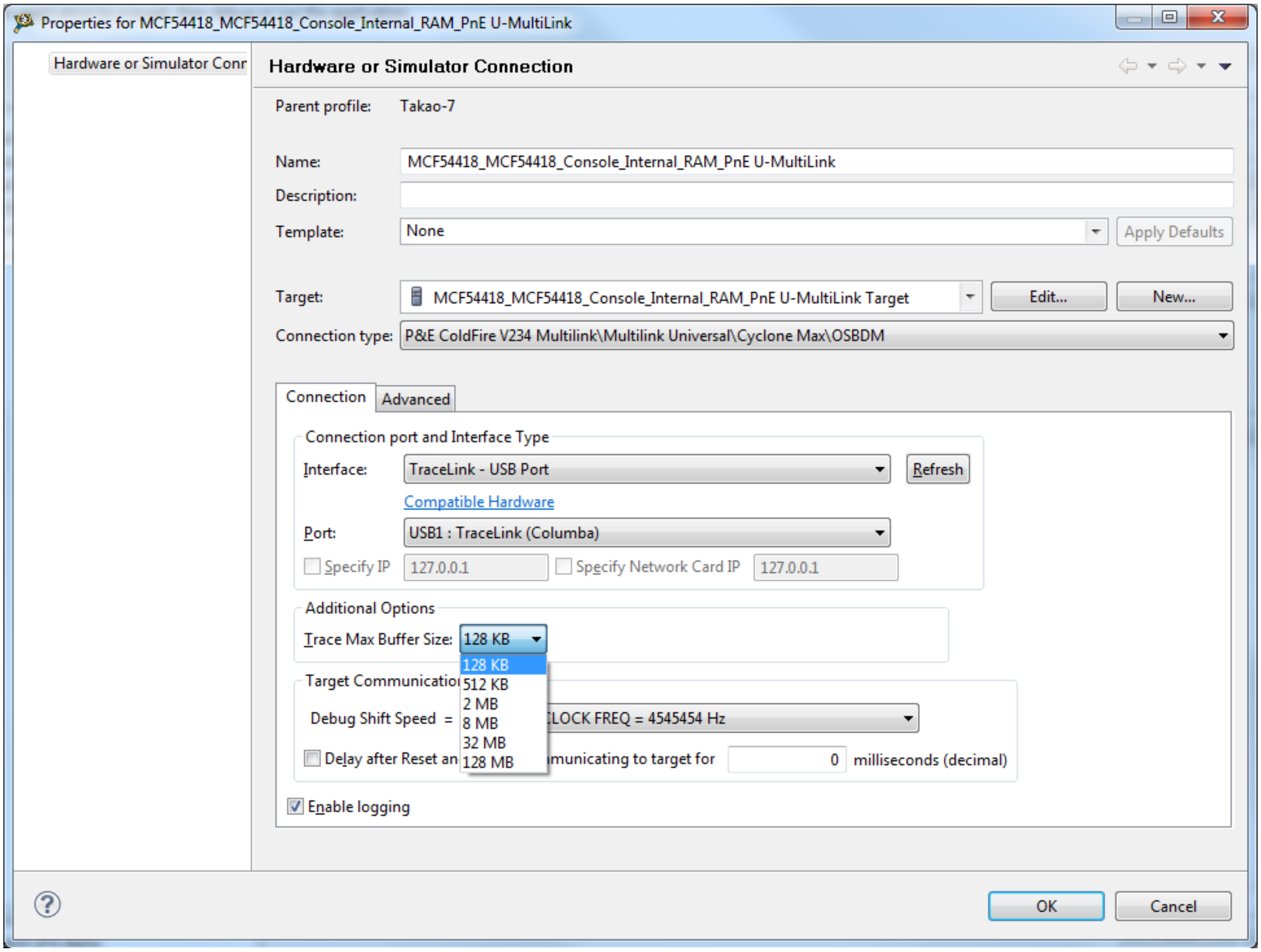
Trace data values:

☐ Read Data

☐ Write Data

Figure 12-15. Trace and Profile Tab

4. Click on the Enable Trace and Profile checkbox.
5. Change the other user settings that fits the users needs.
6. Click on the Main tab and select the correct connection setting from the drop-down box or create a new connection by clicking on the New button.
7. Once the correct connection setting is selected, click on the Edit button. The launch configuration dialog box will appear.
8. When the TraceLink is selected as the interface, the Additional Options will be available. Change the Trace Max Buffer Size as the user sees fit by using the drop-down box.



Properties for MCF54418_MCF54418_Console_Internal_RAM_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MCF54418_MCF54418_Console_Internal_RAM_PnE U-MultiLink

Description:

Template: None

Target: MCF54418_MCF54418_Console_Internal_RAM_PnE U-MultiLink Target

Connection type: P&E ColdFire V234 Multilink\Multilink Universal\Cyclone Max\OSBDM

Connection port and Interface Type

Interface: TraceLink - USB Port

Port: USB1 : TraceLink (Columba)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

Target Communicatio

Debug Shift Speed = CLOCK FREQ = 4545454 Hz

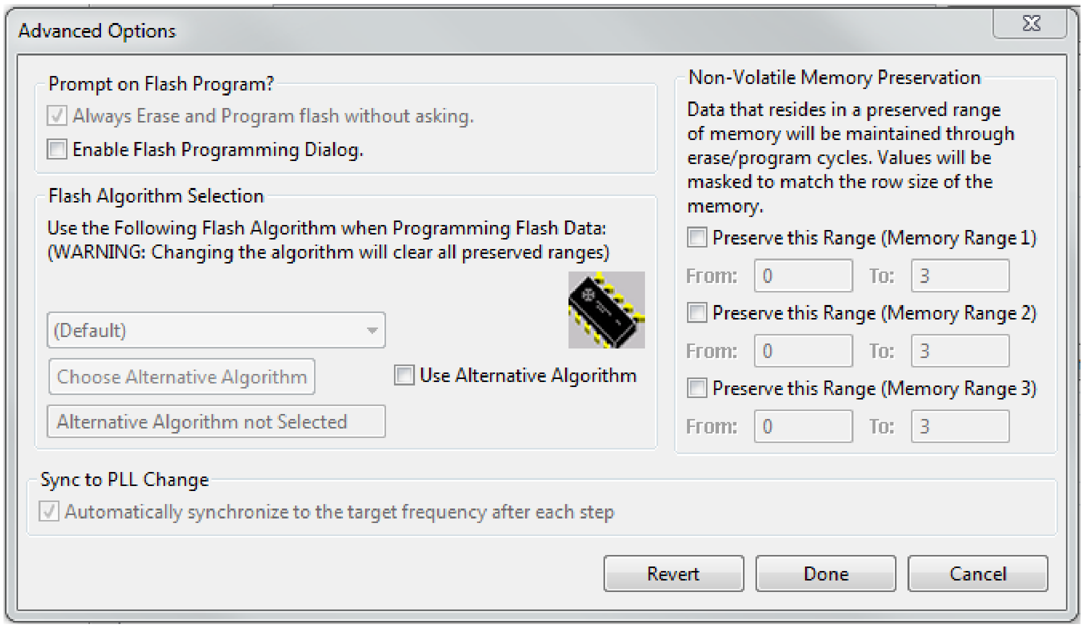
□ Delay after Reset an municating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 12-16. Debugger Settings

12.1.3.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

☐ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

☐ Use Alternative Algorithm

Alternative Algorithm not Selected

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

☐ Preserve this Range (Memory Range 1)

From: 0 To: 3

☐ Preserve this Range (Memory Range 2)

From: 0 To: 3

☐ Preserve this Range (Memory Range 3)

From: 0 To: 3

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

Figure 12-17. Advanced Options Dialog Box

12.1.3.2.1 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

12.1.3.2.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the EEPROM array may optionally be preserved as well).

Ranges that are designated as "preserved" are read before an erase, and reprogrammed immediately afterwards, thereby preserving the data in these ranges. Any attempt to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

12.1.3.2.3 Alternative Algorithm Functionality

Once you create a project for a specific ColdFire V2/3/4 microprocessor, the debugger specifies a default algorithm to use during all Flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the <CW_Install>/MCU/bin/plugins/support/ColdFire/gdi/ P&E directory

However, you can override the default algorithm via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. This feature can be used to select a custom programming algorithm, or select another one of P&E's many programming algorithms for use with a specific project.

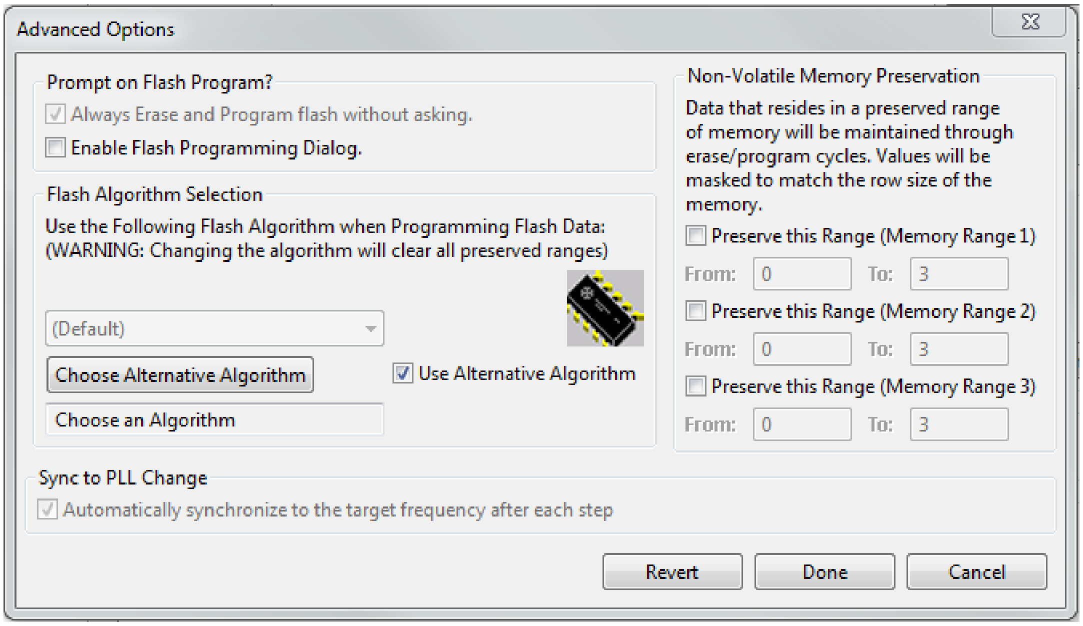
Tip

> Selecting a wrong programming algorithm may damage your device, lead to under/ over programming situations, or simply not program portions of the project file.

Therefore it is recommended to use the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. 1. Check the Use Alternative Algorithm checkbox.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

☐ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

☑ Use Alternative Algorithm

Choose an Algorithm

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

☐ Preserve this Range (Memory Range 1)

From: 0 To: 3

☐ Preserve this Range (Memory Range 2)

From: 0 To: 3

☐ Preserve this Range (Memory Range 3)

From: 0 To: 3

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

Revert | Done | Cancel

Figure 12-18. Advanced Options - Alternative Algorithm Checkbox

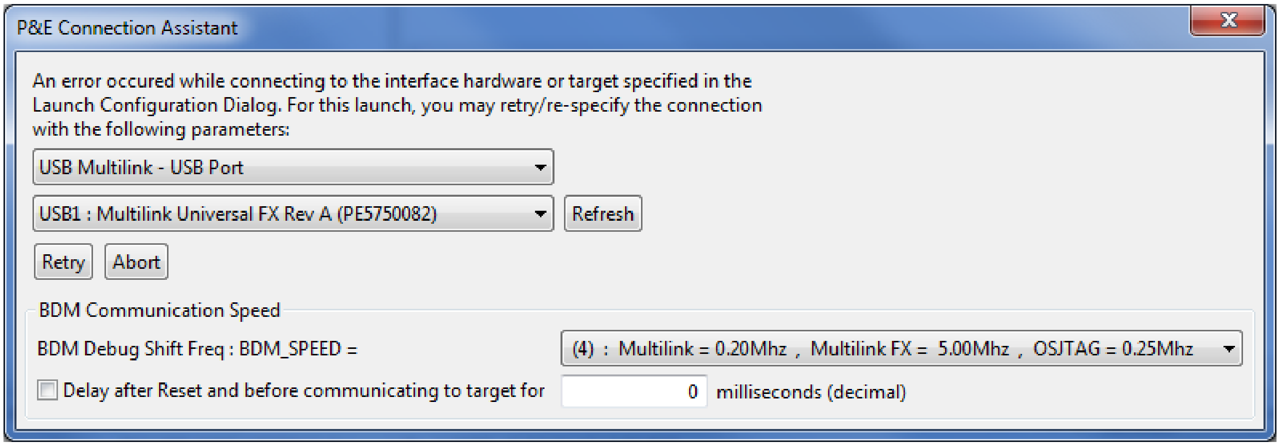
2. 2. Click the Choose Alternative Algorithm button, which lets you browse for an alternative algorithm.
3. 3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the Choose Alternative Algorithm button.

At this point, the current project performs all future Flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

12.1.3.3 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug but CodeWarrior cannot connect to the interface hardware specified in the Launch Configuration dialog box. To edit or change your debugger connection:

1. Choose the P&E device that you are using from the first drop-down menu and click **Refresh**.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the BDM Communication Speed panel to configure the shift frequency and delay. Refer to Changing P&E Connection Settings for more details regarding each setting.
4. Click the Retry button.



P&E Connection Assistant

An error occured while connecting to the interface hardware or target specified in the Launch Configuration Dialog. For this launch, you may retry/re-specify the connection with the following parameters:

USB Multilink - USB Port

USB1 : Multilink Universal FX Rev A (PE5750082)

BDM Communication Speed

BDM Debug Shift Freq : BDM_SPEED = (4) : Multilink = 0.20Mhz , Multilink FX = 5.00Mhz , OSJTAG = 0.25Mhz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

Figure 12-19. CFV2/3/4 Connection Assistant Interface Selected

12.1.4 Active Mode Menu Options

When the microprocessor is connected, the Active Mode menu shows the name of the microprocessor and gives you access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM web page. When the microprocessor is not connected, the menu is not available.

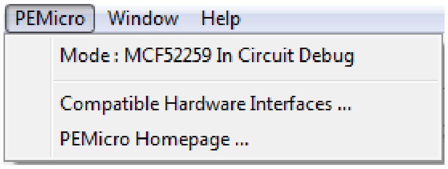

PEMicro
Window
Help
Mode : MCF52259 In Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 12-20. ColdFire Active Mode Menu

Chapter 13
Connections - Qorivva MPC55xx/56xx

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the Qorivva MPC55xx/56xx target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

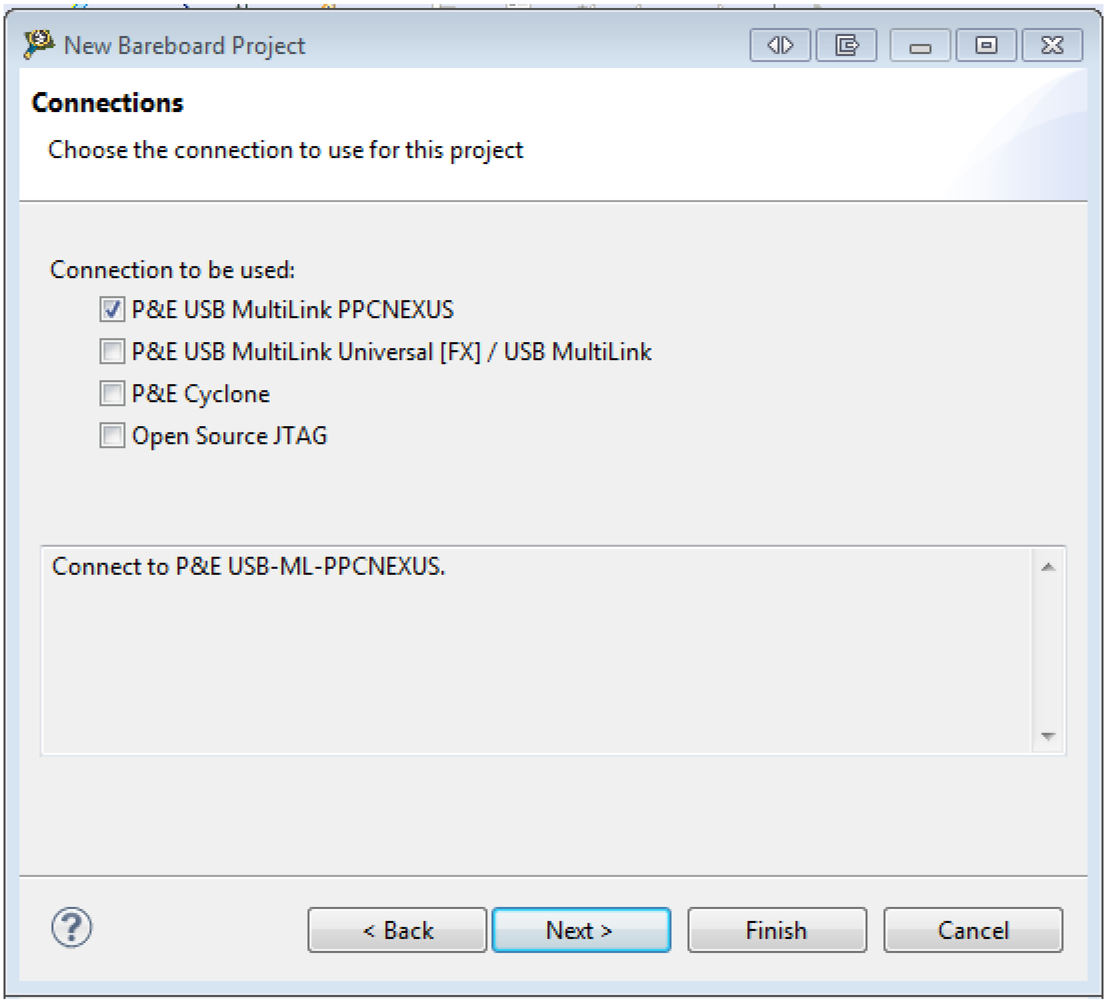
The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with the Qorivva MPC55xx/56xx device family.

13.1 P&E Hardware Interface Connection for Qorivva

This section describes Qorivva Connection options. The QORIVVA P&E Connection setting permits a connection to QORIVVA Freescale devices via P&E Multilink, Cyclone (including Cyclone MAX), and OSJTAG hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

13.1.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.



Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink PPCNEXUS
- □ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ Open Source JTAG

Connect to P&E USB-ML-PPCNEXUS.

Figure 13-1. New project wizard

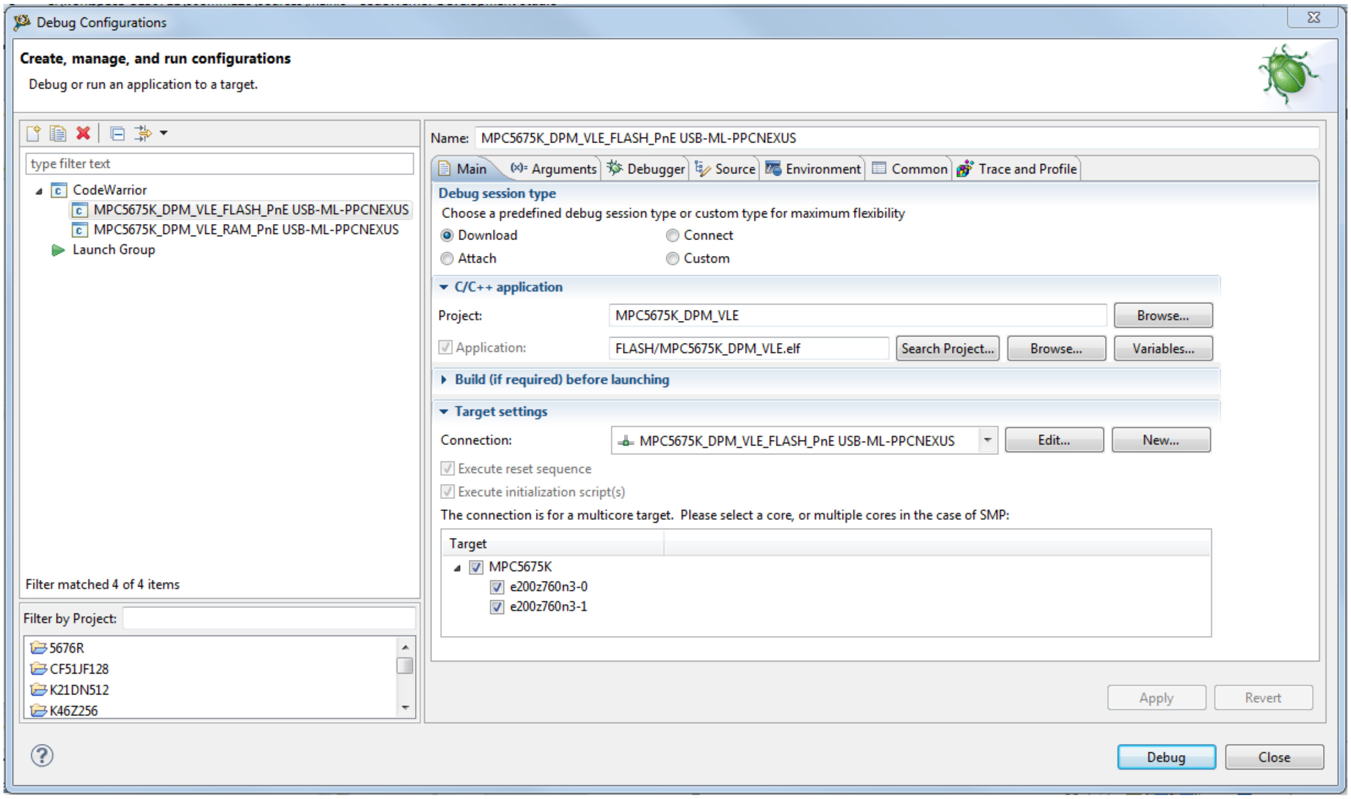
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

13.1.2 Launch Configurations Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project type you would like to set the launch configurations.
3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - MPC5675K_DPM_VLE_FLASH_PnE USB-ML-PPCNEXUS
  - MPC5675K_DPM_VLE_RAM_PnE USB-ML-PPCNEXUS
- Launch Group

Filter matched 4 of 4 items

Filter by Project:

- 5676R
- CF51JF128
- K21DN512
- K46Z256

Name: MPC5675K_DPM_VLE_FLASH_PnE USB-ML-PPCNEXUS

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: MPC5675K_DPM_VLE

☑ Application: FLASH/MPC5675K_DPM_VLE.elf

Build (if required) before launching

Target settings

Connection: MPC5675K_DPM_VLE_FLASH_PnE USB-ML-PPCNEXUS

☑ Execute reset sequence

☑ Execute initialization script(s)

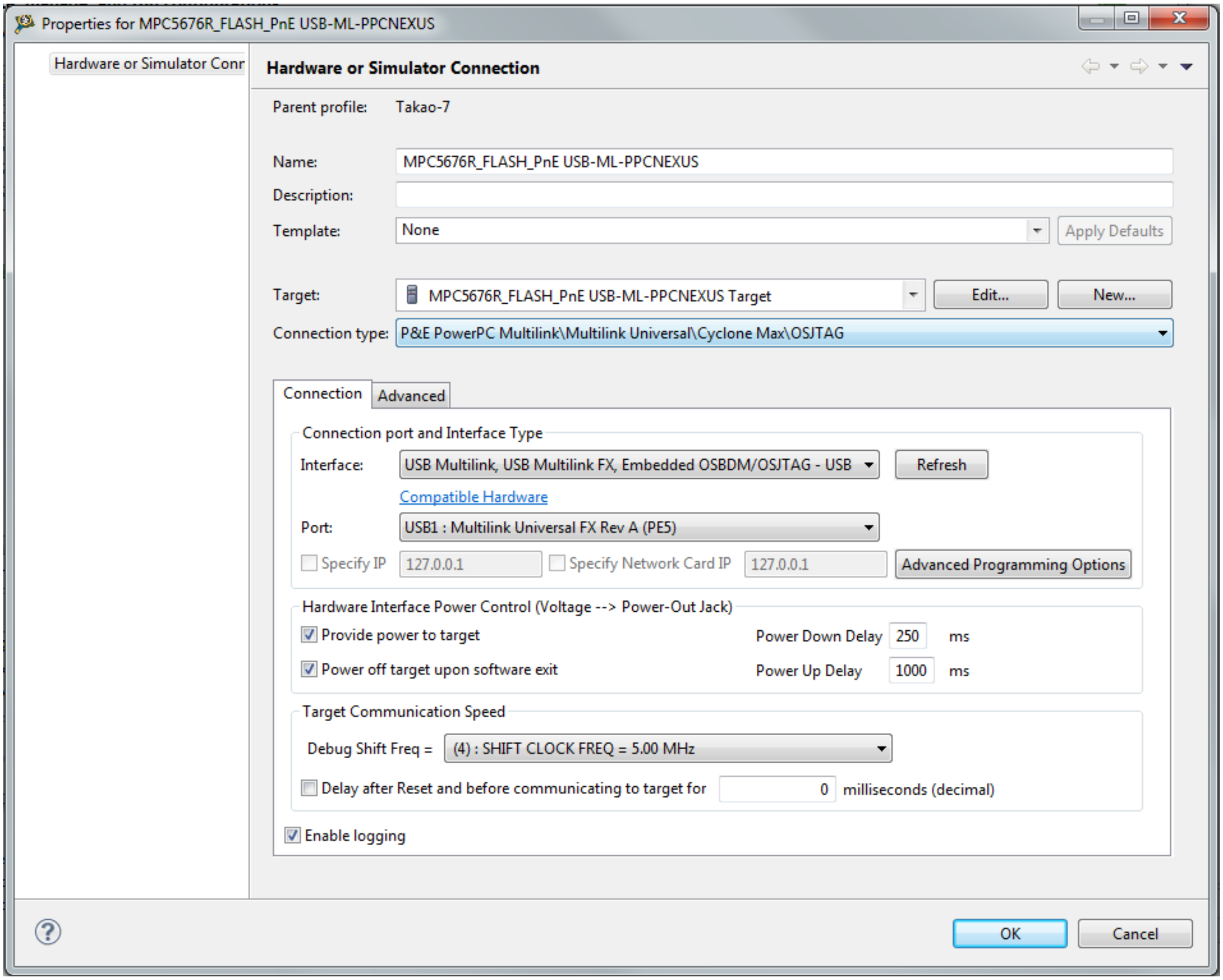
The connection is for a multicore target. Please select a core, or multiple cores in the case of SMP:

Target

- ☑ MPC5675K
  - ☑ e200z760n3-0
  - ☑ e200z760n3-1

Figure 13-2. Debug Configuration Dialog Box

4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear..
5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Properties for MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Apply Defaults

Target: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS Target

Edit... New...

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Refresh

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Advanced Programming Options

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

Power Down Delay 250 ms

☑ Power off target upon software exit

Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

OK Cancel

Figure 13-3. Launch Configuration Dialog Box

13.1.3 Connection Options

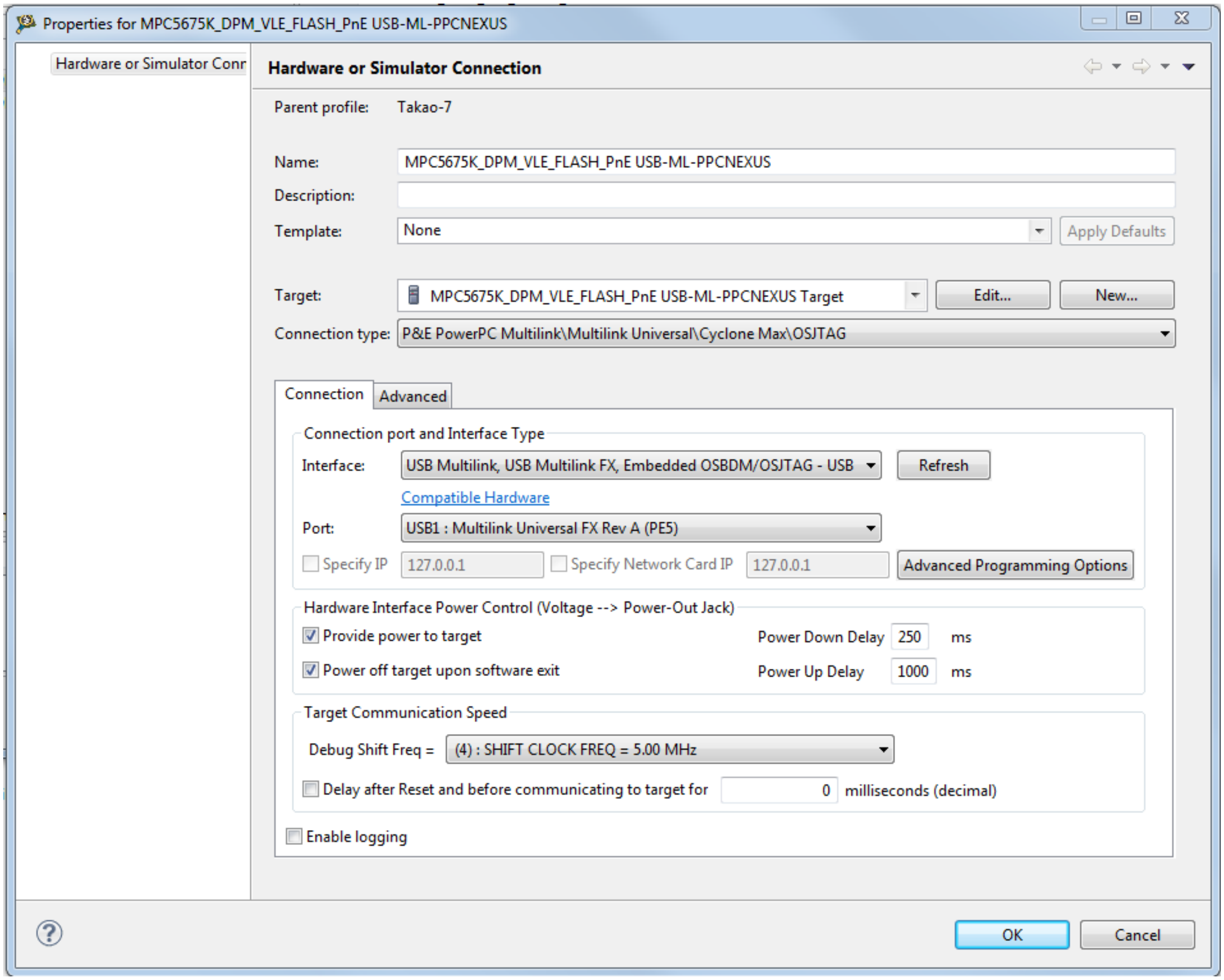
This topic describes all P&E Multilink/Cyclone /OSJTAG connection options, which are common to all P&E USB Multilink Universal [FX]/USB Multilink, P&E Cyclone Serial , P&E Cyclone USB , P&E Cyclone Ethernet , and Open Source JTAG connections.

The options include:

- Changing P&E Connections Settings
- Connection Assistant

13.1.3.1 Changing P&E Connections Settings

All connection settings for P&E hardware interfaces are configured in the Launch Configurations dialog box.



Properties for MPC5675K_DPM_VLE_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5675K_DPM_VLE_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Target: MPC5675K_DPM_VLE_FLASH_PnE USB-ML-PPCNEXUS Target

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 13-4. P&E QORIVVA Launch Configuration Dialog Box

The following table describes the options of the view.

Table 13-1. Connection Parameter Options for P&E Multilink/Cyclone /OSJTAG

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port • Cyclone - Serial Port • Cyclone - USB Port • Cyclone - Ethernet Port

Table continues on the next page...

Table 13-1. Connection Parameter Options for P&E Multilink/Cyclone /OSJTAG (continued)

Option	Description
	NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.
Advanced Programming Options	The Advanced Programming Options button brings up a dialog that provides you with options to configure the flash programming procedure.
Specify IP (Cyclone Ethernet only)	Use this option to specify the IP address of a Cyclone outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port dropdown box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Provide power to target (USB Multilink Universal FX only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Power down delay (USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Debug Shift Freq.	This option lets you to set the debug shift clock speed of P&E's interfaces. This integer value may be used to determine the speed of communications according to the following equations: • Cyclone : (50000000/(2*N+5)) Hz • USB-ML-PPCNEXUS: (1000000/(N+1)) Hz • USB Multilink Universal: (1000000/(N+1)) Hz • USB ML Universal FX : (25000000/(N+1)) Hz OSJTAG : Fixed Frequency The value n should be between 0 and 31. This shift clock takes effect after the commands in the top of the programming algorithm are executed so that these commands can increase the target frequency and allow a faster shift clock. This clock can't generally exceed a div 4 of the processor bus frequency.

Table continues on the next page...

Table 13-1. Connection Parameter Options for P&E Multilink/Cyclone /OSJTAG (continued)

Option	Description
Delay After Reset	Specifies a delay after the programmer resets the target that we check to see if the part has properly gone into background debug mode. This is useful if the target has a reset driver which hold the MCU in reset after the programmer releases the reset line. The n value is a delay in milliseconds.

To change P&E Connections settings, perform these steps:

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

   NOTE

   It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

   The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection of which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Hardware or Simulator Connection panel appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E PowerPC Multilink\Multilink Universal\Cyclone \OSJTAG. The P&E connections settings will appear below.
8. Click Refresh to scan valid interface and port.

   Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

NOTE

> The port displayed may vary depending on the interface. For example, if you select interface as Cyclone - Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Target Communication Speed group.

NOTE

> This group will be enabled for the Tracelink and USB Multilink Universal FX interfaces only. For USB Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

- Select a Debug Shift Frequency.

  NOTE

  > If you select a fast speed, there may be scenarios where you may have difficulty communicating with your target device. You may want to experimentally select a speed that is fast and yet have a good communication with your target.

- Click on the Delay after Reset checkbox and enter the desired delay (in milliseconds) in the text box.

  This option specifies the time interval to wait between resetting and communicating the target device.

- Click the OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.

Click on the Close button to close the Debug Configuration dialog box.

13.1.3.1.1 P&E Hardware Interface Connection- Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]/USB Multilink
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- Open Source JTAG

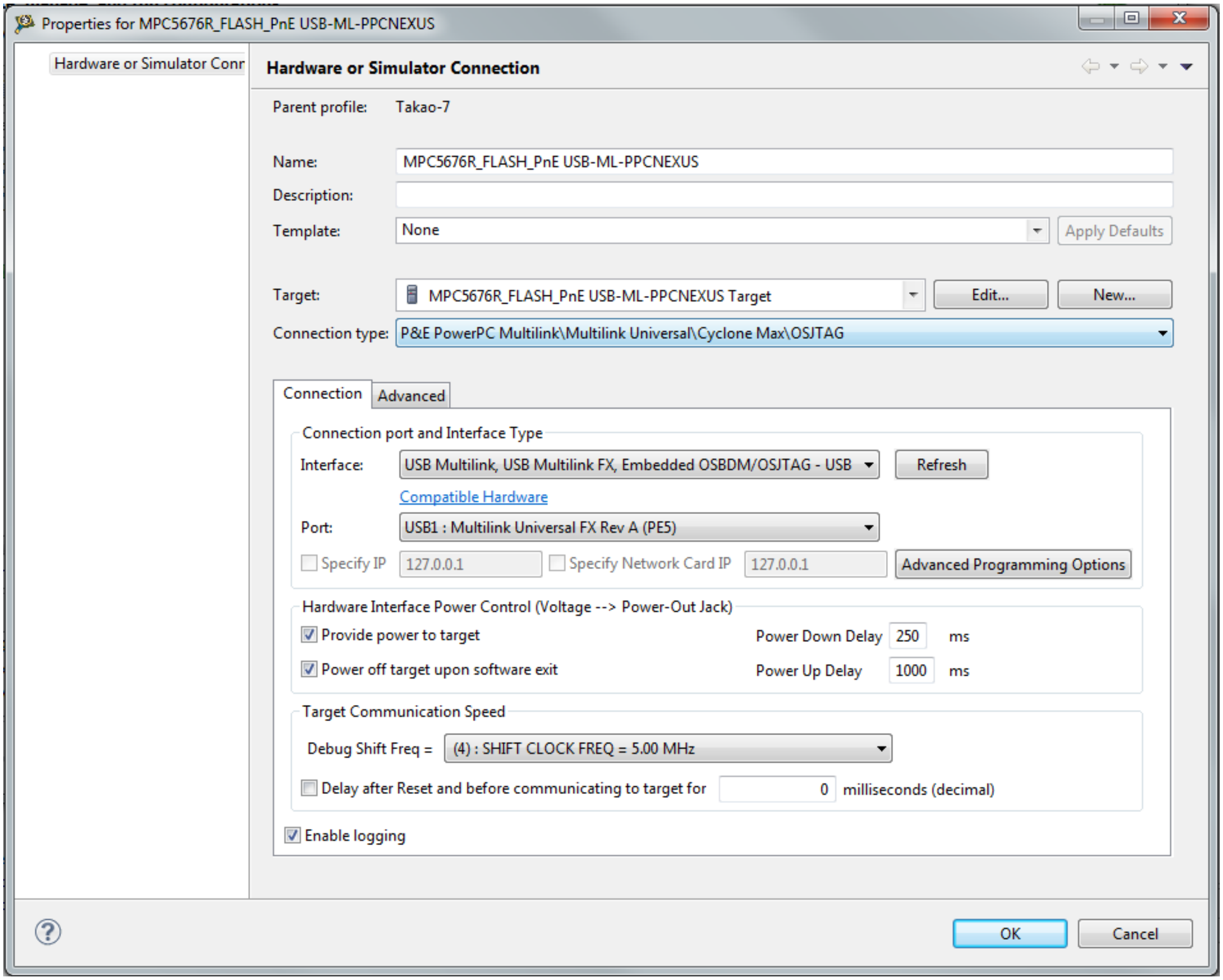
13.1.3.1.1.1 P&E USB Multilink Universal [FX]/USB Multilink

The P&E USB Multilink Universal [FX]/USB Multilink Connection setting permits a connection to USB Multilink devices, which include the P&E Qorivva Multilink, USB Multilink Universal, and the USB Multilink Universal FX. P&E USB Multilink Universal [FX]/USB Multilink mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

13.1.3.1.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E PowerPC Multilink/Multilink Universal/Cyclone Max /OSJTAG so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Target: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS Target

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 13-5. P&E's Launch Configuration Dialog Box

To use P&E's USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

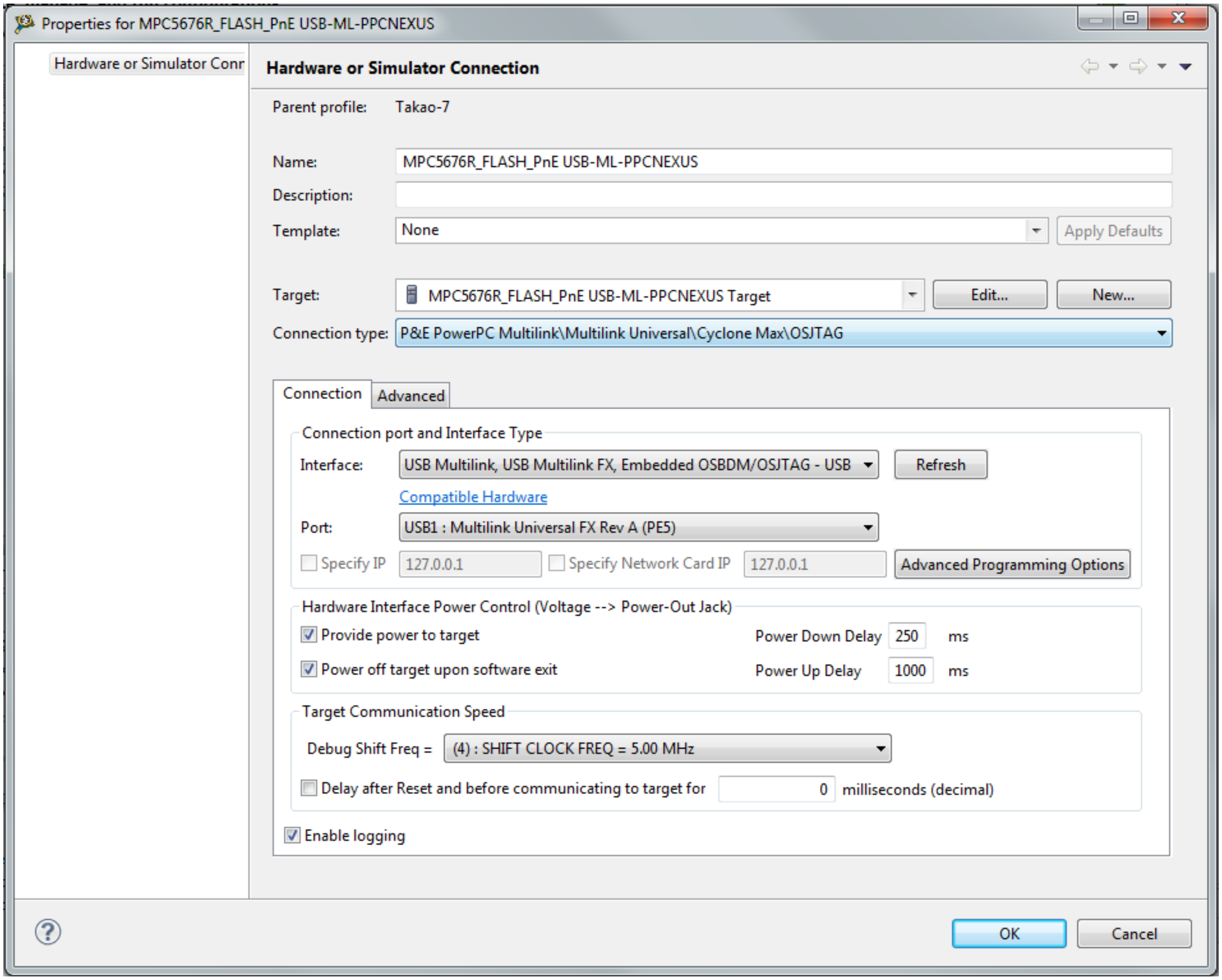
13.1.3.1.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

13.1.3.1.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E PowerPC Multilink/Multilink Universal/Cyclone Max/OSJTAG so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Target: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS Target

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 13-6. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone - Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

> The Cyclone MAX does not support the ability to provide power to the target.

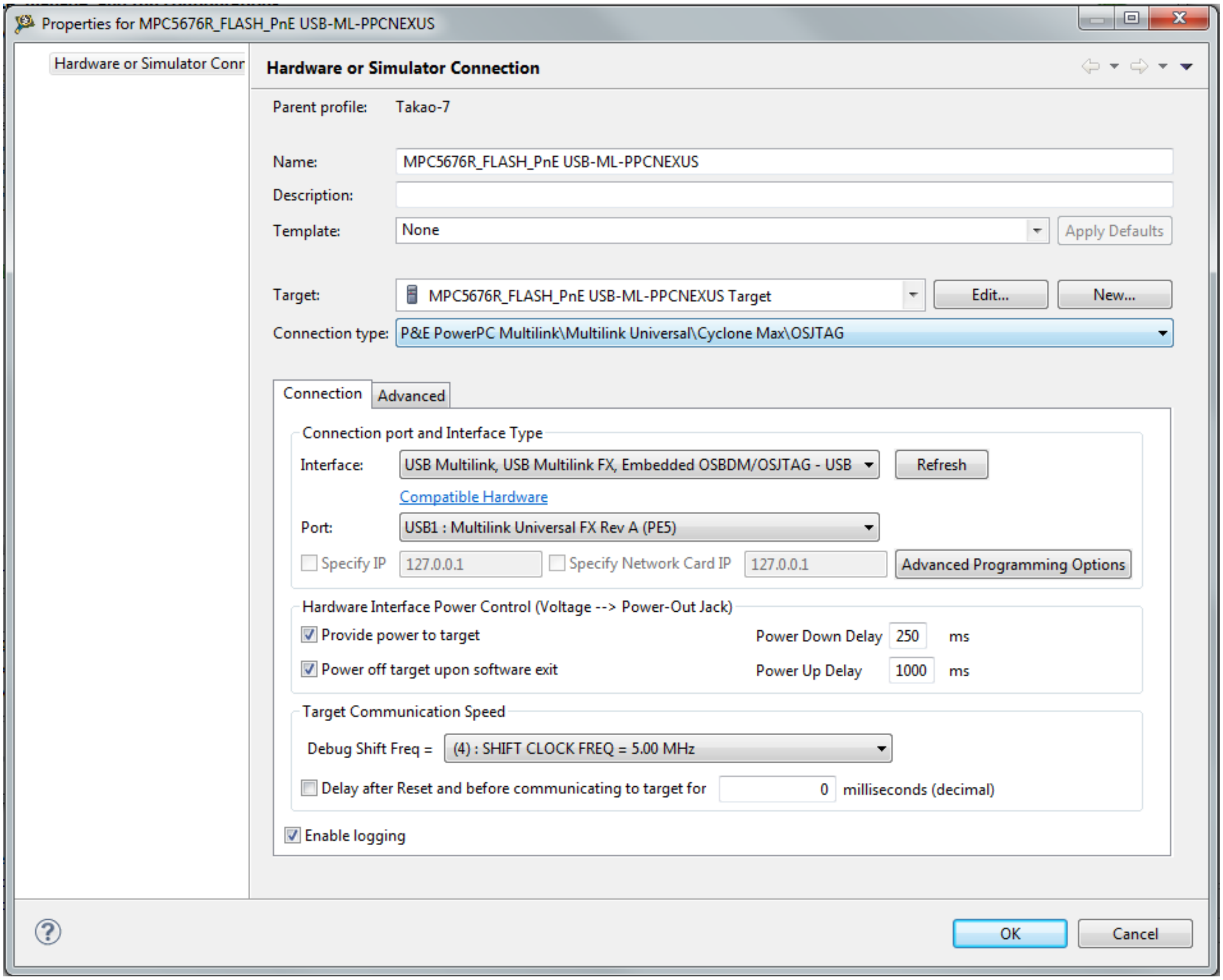
13.1.3.1.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

13.1.3.1.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E PowerPC Multilink/Multilink Universal/Cyclone Max/OSJTAG so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Target: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS Target

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 13-7. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

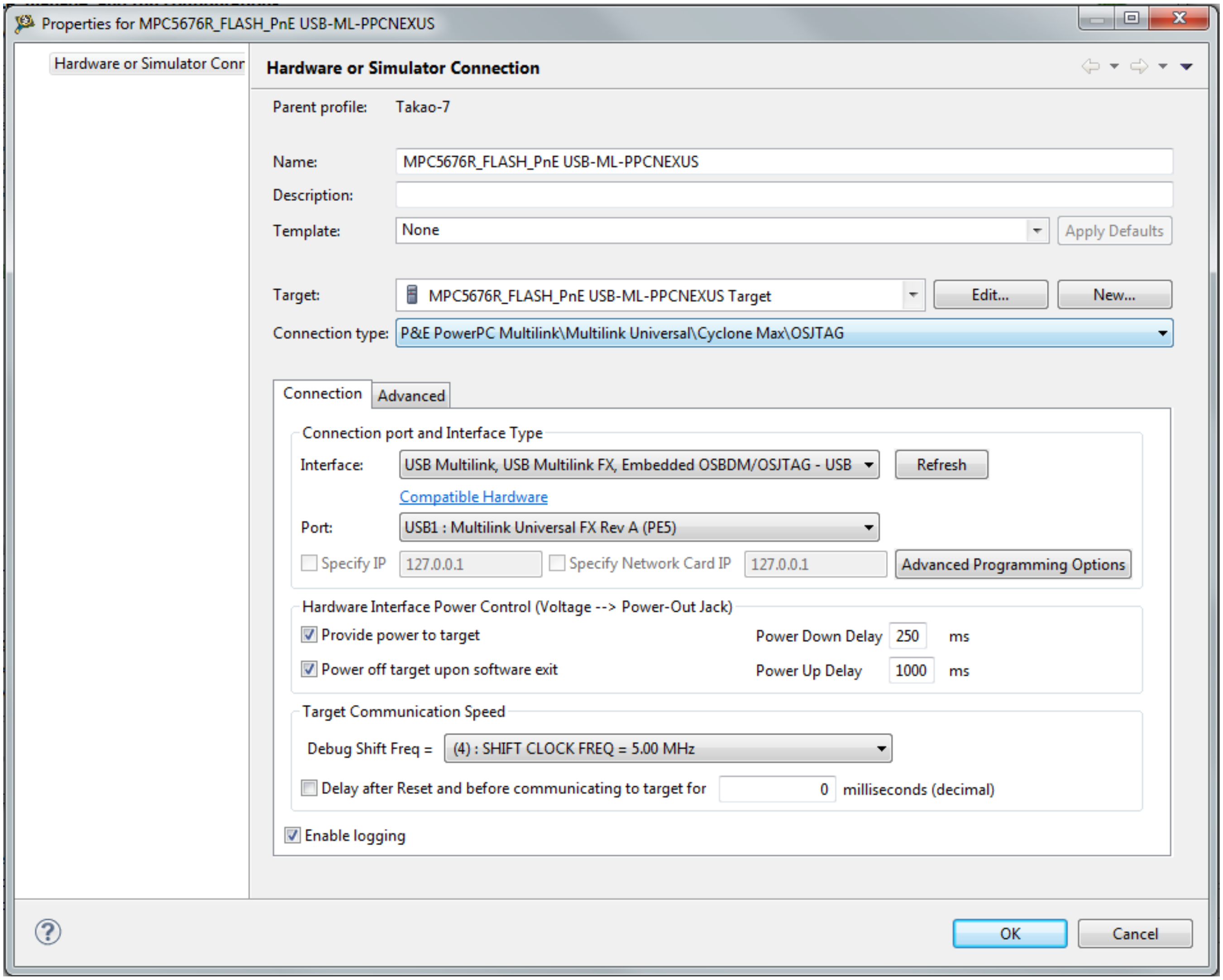
13.1.3.1.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

13.1.3.1.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E PowerPC Multilink/Multilink Universal/Cyclone Max/OSJTAG so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Target: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS Target

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 13-8. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone - Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target

.

13.1.3.1.1.5 Open Source JTAG

Freescale supplies certain development boards with an integrated debug circuit based on Open Source JTAG. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB QORIVVA Multilink. The development board also derives its power from the USB Bus.

The Open Source JTAG circuit design (OSJTAG-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

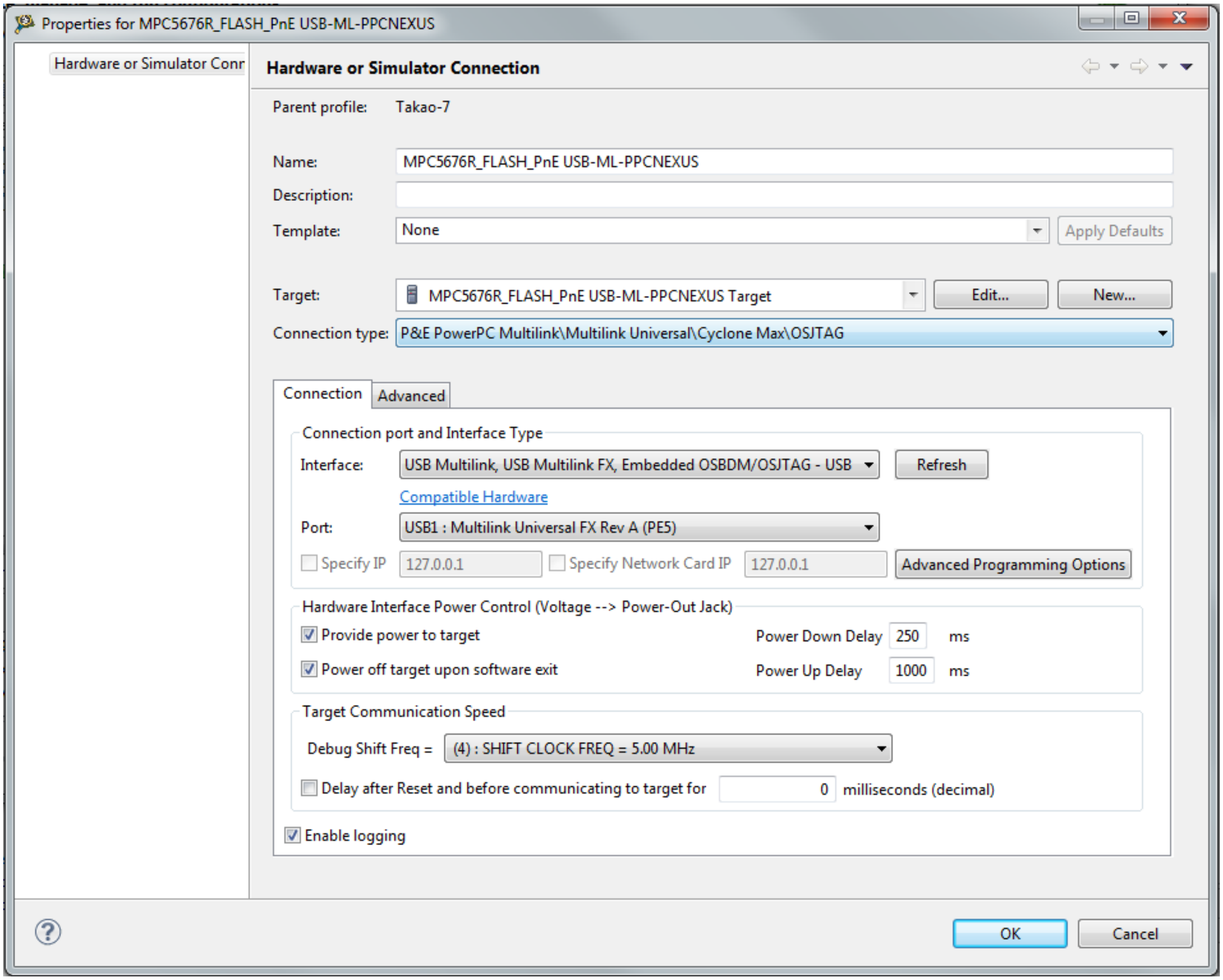
Integration with CodeWarrior is handled via the "P&E Open Source TAG" connection. P&E has integrated the Open Source BDM support into the same connection that supports both the USB ColdFire Multilink and the Cyclone . All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSJTAG (albeit at a lower data rate).

The Open Source JTAG Connection setting permits a connection to Open Source JTAG devices. Open Source JTAG mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

13.1.3.1.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E PowerPC Multilink/Multilink Universal/Cyclone Max/OSJTAG so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS

Description:

Template: None

Target: MPC5676R_FLASH_PnE USB-ML-PPCNEXUS Target

Connection type: P&E PowerPC Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target   Power Down Delay 250 ms

☑ Power off target upon software exit   Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

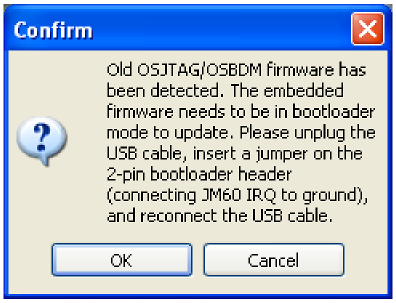
☑ Enable logging

Figure 13-9. P&E's Launch Configuration Dialog Box

To use Open Source JTAG, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

13.1.3.1.1.5.2 OSJTAG Firmware Update

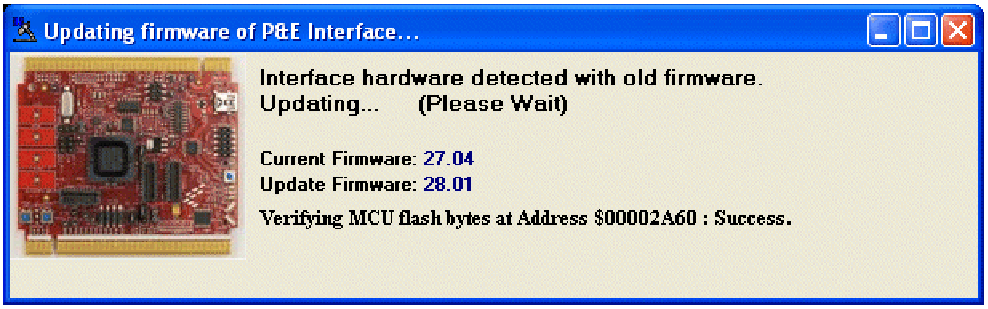
All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSJTAG hardware interfaces. Whenever an OSJTAG-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSJTAG firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.



Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 13-10. Old OSJTAG Firmware Detected

To update the firmware, the OSJTAG device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSJTAG-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSJTAG device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSJTAG device and click on the OK button. If done correctly, the automatic firmware update will occur.



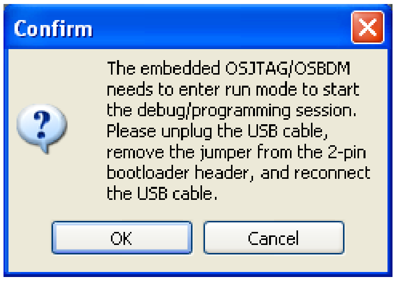
Interface hardware detected with old firmware.
Updating... (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01

Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 13-11. OSJTAG Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSJTAG device must exit Bootloader mode and enter into Run mode.



The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

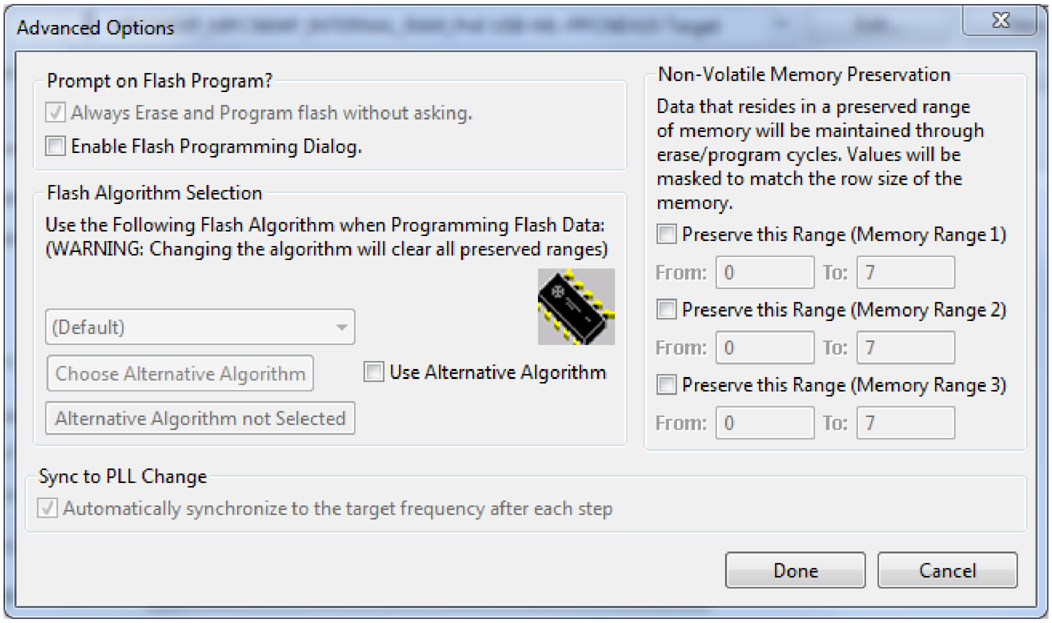
Figure 13-12. Start OSJTAG Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSJTAG device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSJTAG firmware. If for any reason you experience difficulty performing OSJTAG firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

13.1.3.1.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

□ Use Alternative Algorithm

Alternative Algorithm not Selected

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 0 To: 7

□ Preserve this Range (Memory Range 2)

From: 0 To: 7

□ Preserve this Range (Memory Range 3)

From: 0 To: 7

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

Figure 13-13. Advanced Options Dialog Box

13.1.3.1.2.1 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

13.1.3.1.2.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase, and reprogrammed immediately afterwards, thereby preserving the data in these ranges. Any attempt to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

13.1.3.1.2.3 Alternative Algorithm Functionality

Once you create a project for a specific Qorivva MPC55xx/56xx microprocessor, the debugger specifies a default algorithm to use during all Flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the <CW_Install>/MCU/bin/plugins/support/EPPC/gdi/P&E directory

However, you can override the default algorithm via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. This feature can be used to select a custom programming algorithm, or select another one of P&E's many programming algorithms for use with a specific project.

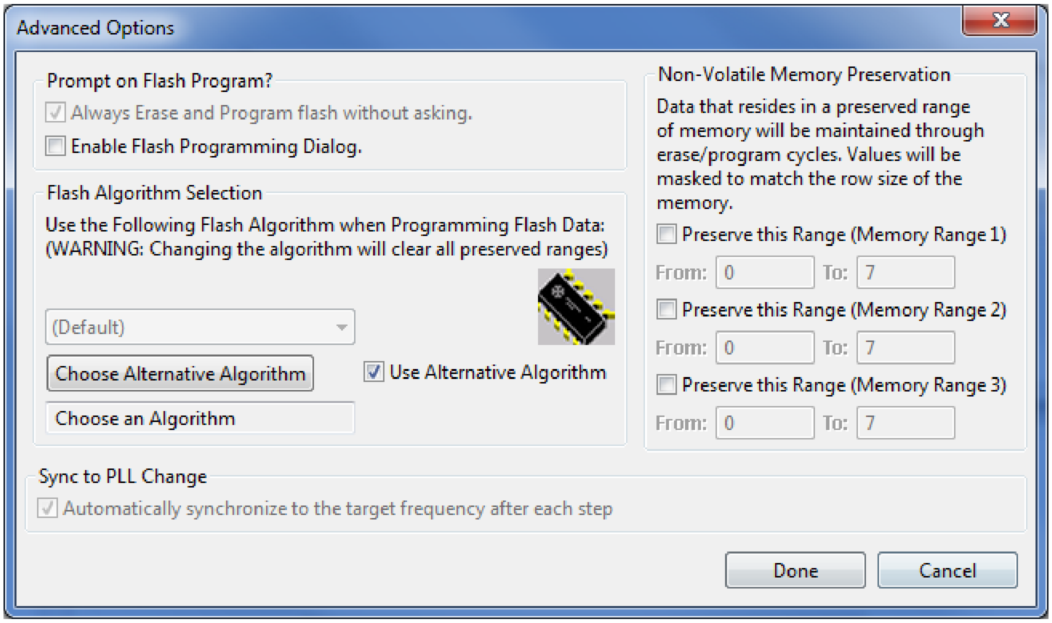
Tip

> Selecting a wrong programming algorithm may damage your device, lead to under/ over programming situations, or simply not program portions of the project file.

Therefore it is recommended to use the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. Check the Use Alternative Algorithm checkbox.



Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

☑ Use Alternative Algorithm

Choose an Algorithm

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 0 To: 7

□ Preserve this Range (Memory Range 2)

From: 0 To: 7

□ Preserve this Range (Memory Range 3)

From: 0 To: 7

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

Figure 13-14. Advanced Options - Alternative Algorithm Checkbox

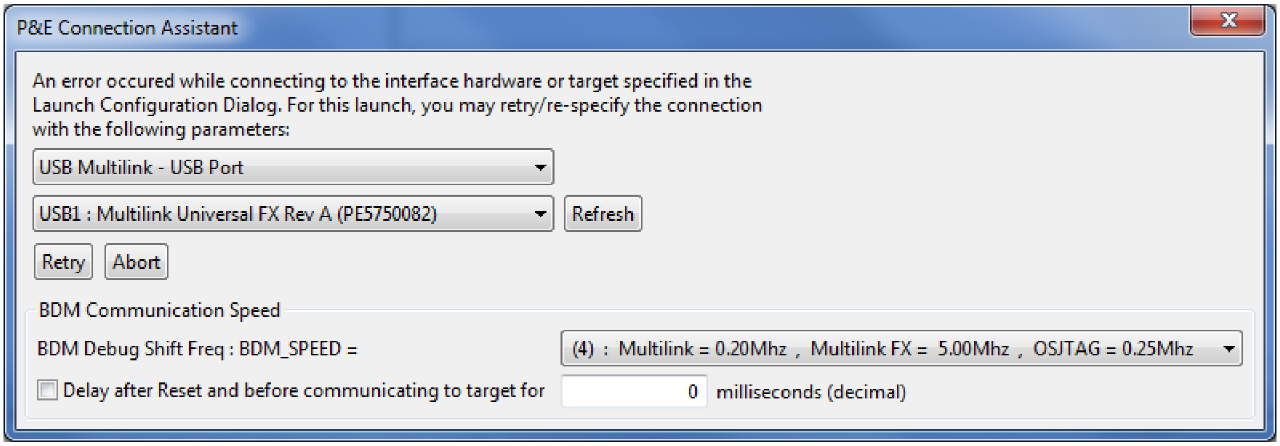
2. Click the Choose Alternative Algorithm button, which lets you browse for an alternative algorithm.
3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the Choose Alternative Algorithm button.

At this point, the current project performs all future Flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

13.1.3.2 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug but CodeWarrior cannot connect to the interface hardware specified in the Launch Configuration dialog box. To select the P&E Multilink/Cyclone /OSJTAG as your debugger connection:

1. Choose the P&E device that you are using from the first drop-down menu and click Refresh.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the BDM Communication Speed panel to configure the shift frequency and delay. Refer to Changing P&E Connections Settings for more details regarding each setting.
4. Click the Retry button.



P&E Connection Assistant

An error occured while connecting to the interface hardware or target specified in the Launch Configuration Dialog. For this launch, you may retry/re-specify the connection with the following parameters:

USB Multilink - USB Port

USB1 : Multilink Universal FX Rev A (PE5750082)

BDM Communication Speed

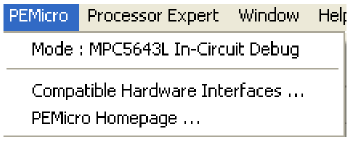
BDM Debug Shift Freq : BDM_SPEED = (4) : Multilink = 0.20Mhz , Multilink FX = 5.00Mhz , OSJTAG = 0.25Mhz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

Figure 13-15. Connection Assistant Interface Selected

13.1.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.



PEMicro
Processor Expert
Window
Help
Mode : MPC5643L In-Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 13-16. Additional Connection Menu Options

Chapter 14
Connections — Kinetis Architecture

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the Kinetis target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

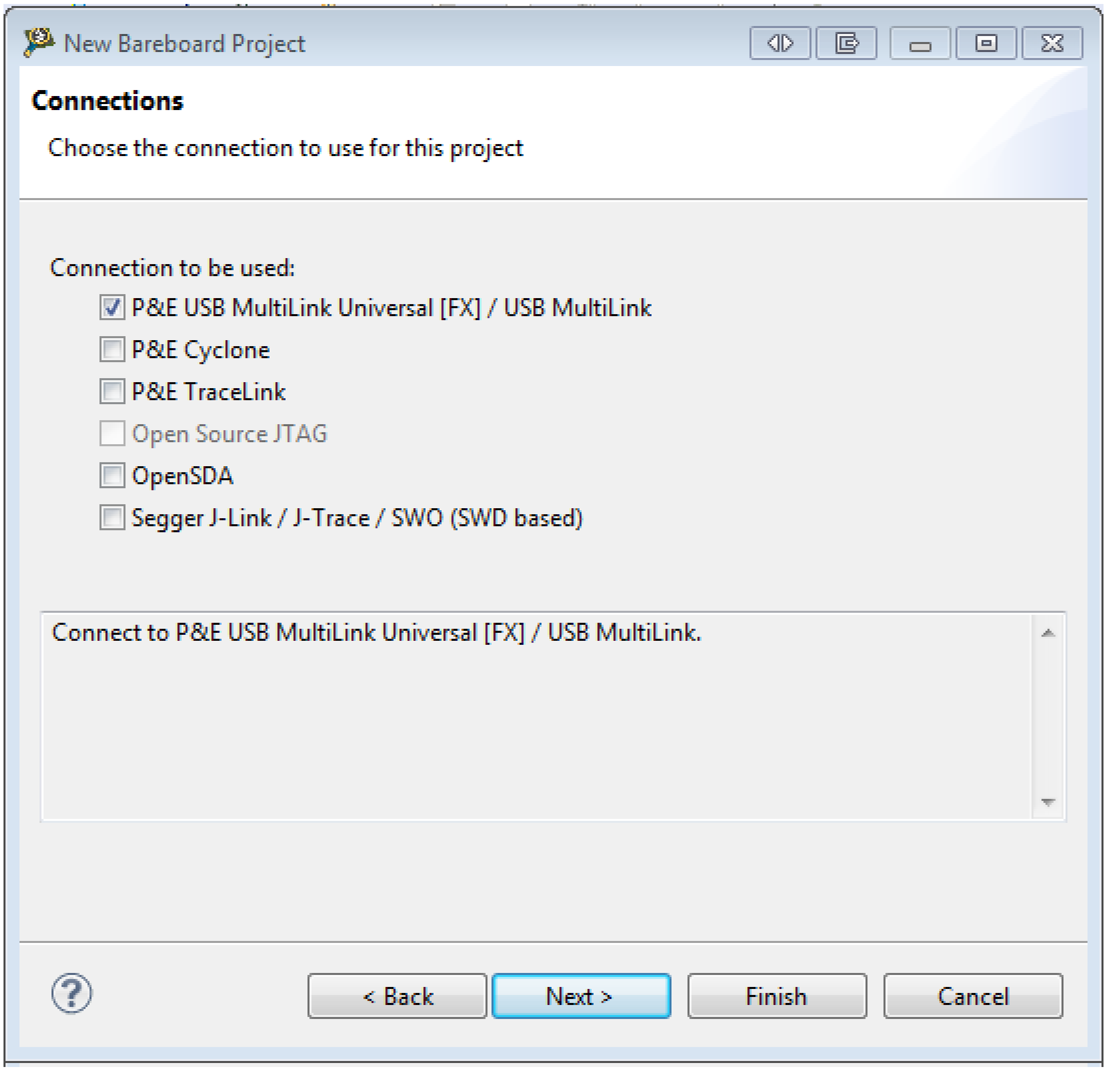
The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with the Kinetis device family.

14.1 P&E Hardware Interface Connection for Kinetis

This section describes Kinetis K-, L-, E-, and M-Series P&E Connection options. The Kinetis P&E Connection setting permits a connection to Kinetis Freescale devices via P&E Multilink, Cyclone , Tracelink, OSJTAG, and OpenSDA hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

14.1.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.



Connections

Choose the connection to use for this project

Connection to be used:

☑ P&E USB MultiLink Universal [FX] / USB MultiLink
□ P&E Cyclone
□ P&E TraceLink
□ Open Source JTAG
□ OpenSDA
□ Segger J-Link / J-Trace / SWO (SWD based)

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 14-1. New project wizard

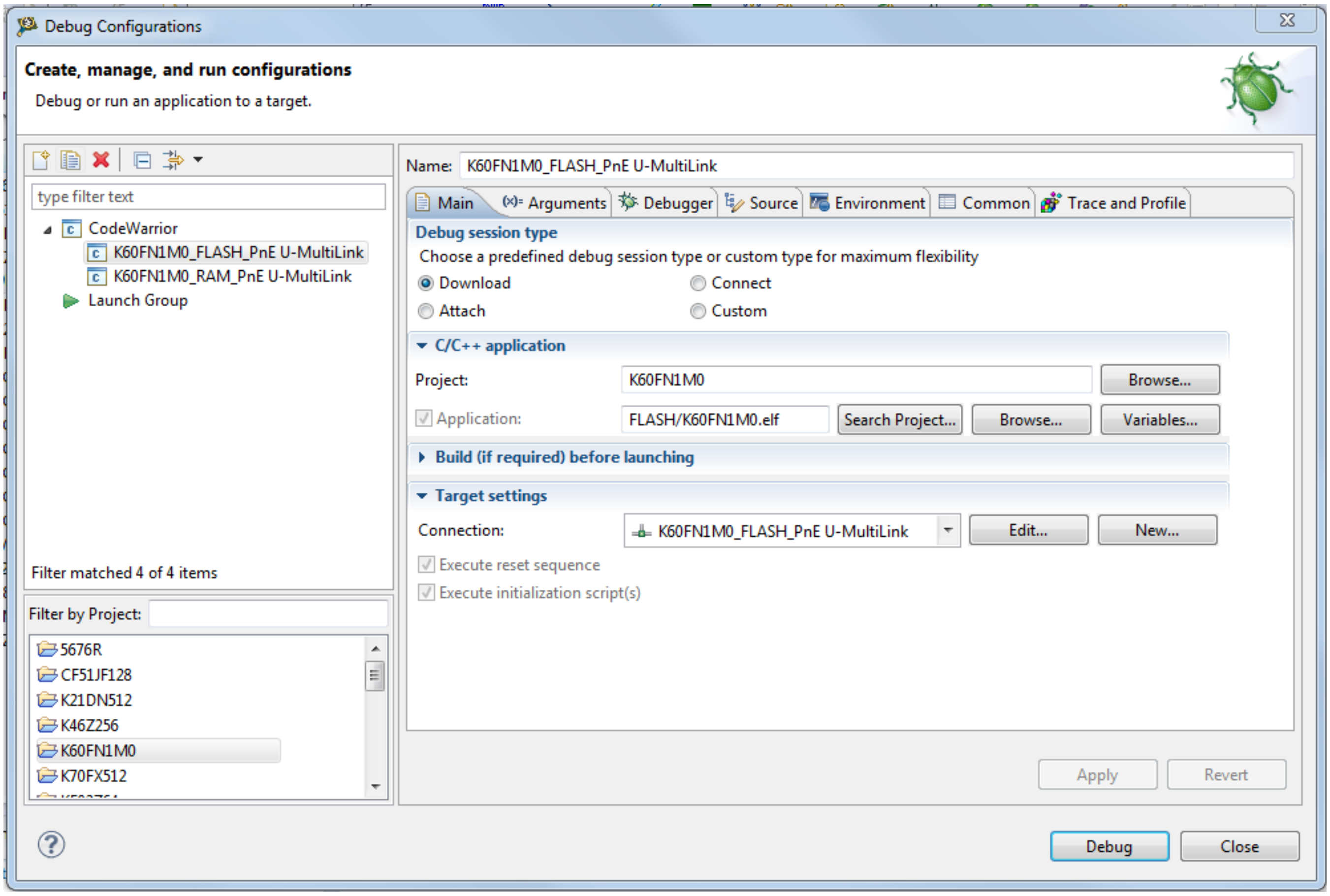
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

14.1.2 Launch Configurations Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project type you would like to set the launch configurations.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - K60FN1M0_FLASH_PnE U-MultiLink
  - K60FN1M0_RAM_PnE U-MultiLink
- Launch Group

Filter matched 4 of 4 items

Filter by Project:

- 5676R
- CF51JF128
- K21DN512
- K46Z256
- K60FN1M0
- K70FX512

Name: K60FN1M0_FLASH_PnE U-MultiLink

Main | Arguments | Debugger | Source | Environment | Common | Trace and Profile

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: K60FN1M0 [Browse...]

☑ Application: FLASH/K60FN1M0.elf [Search Project...] [Browse...] [Variables...]

Build (if required) before launching

Target settings

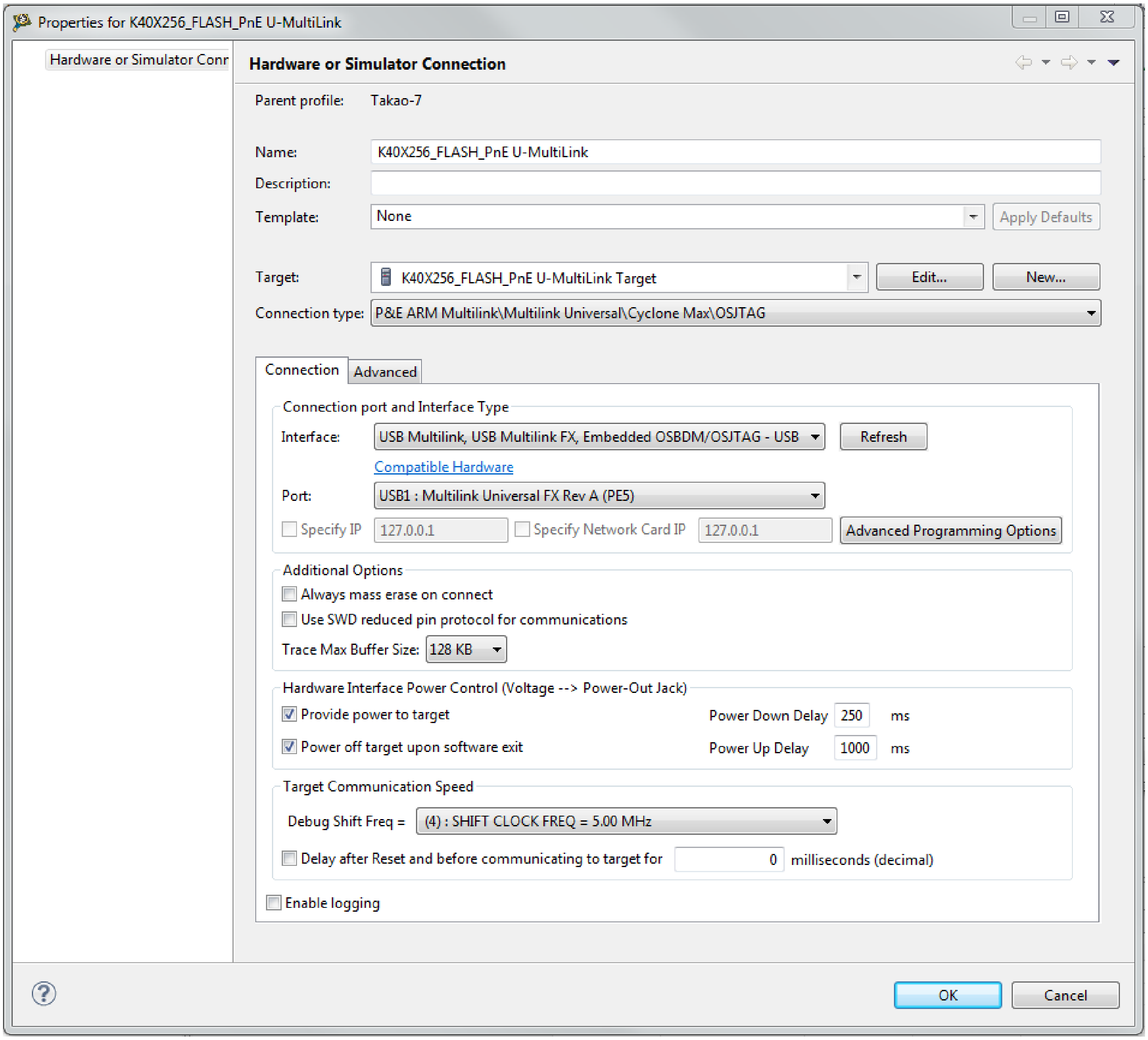
Connection: K60FN1M0_FLASH_PnE U-MultiLink [Edit...] [New...]

☑ Execute reset sequence

☑ Execute initialization script(s)

Figure 14-2. Debug Configuration Dialog Box

3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear.
5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 14-3. Launch Configuration Dialog Box

14.1.3 Connection Options

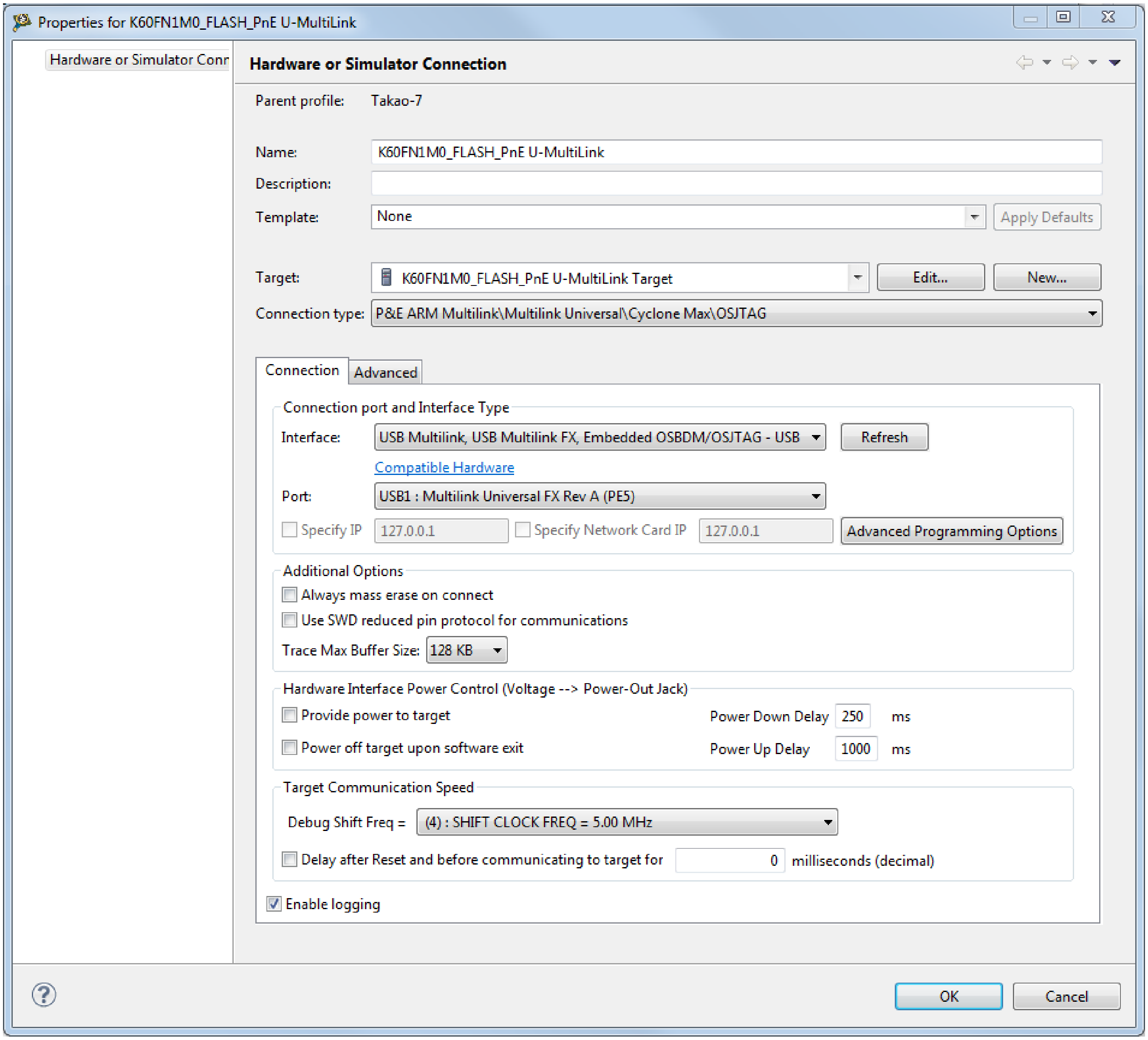
This topic describes all P&E MultiLink/Cyclone /Tracelink/OSJTAG/OpenSDA connection options, which are common to all P&E USB MultiLink Universal [FX], P&E Cyclone Serial , P&E Cyclone USB , P&E Cyclone Ethernet , P&E TraceLink USB, P&E TraceLink Ethernet, Open Source JTAG, and OpenSDA connections.

The options include:

- Changing P&E Connections Settings
- Connection Assistant

14.1.3.1 Changing P&E Connections Settings

All connection settings for P&E hardware interfaces are configured in the Launch Configurations dialog box.



Properties for K60FN1M0_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K60FN1M0_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K60FN1M0_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

□ Provide power to target — Power Down Delay 250 ms

□ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

Figure 14-4. P&E Kinetis Launch Configuration Dialog Box

The following table describes the options for this dialog box.

Table 14-1. Tool Settings - Disable user messages Options

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB Multilink Universal [FX] - USB Port NOTE: The USB Multilink Universal and the USB Multilink Universal FX can conveniently support all Freescale architectures found in the current CodeWarrior 10 version • Cyclone - Serial Port • Cyclone - USB Port • Cyclone - Ethernet Port • TraceLink - USB Port • TraceLink - Ethernet Port • OSJTAG • OpenSDA NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.
Advanced Programming Options	The Advanced Programming Options button brings up a dialog that provides you with options to configure the flash programming procedure.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Specify IP (Cyclone Ethernet and TraceLink Ethernet only)	Use this option to specify the IP address of a hardware interface outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port dropdown box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone Ethernet and TraceLink Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Provide power to target (USB Multilink Universal FX and TraceLink only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (USB Multilink Universal FX and TraceLink only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Regulator Output Voltage (Tracelink only)	This option adjusts the output voltage that powers the hardware target. Select a voltage value from this option's list box. **CAUTION:** An improper voltage setting can damage the board.

Table continues on the next page...

Table 14-1. Tool Settings - Disable user messages Options (continued)

Option	Description
Power down delay (USB Multilink Universal FX and TraceLink only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (USB Multilink Universal FX and TraceLink only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Debug Shift Freq.	This option lets you to set the debug shift clock speed of P&E's interfaces. This integer value may be used to determine the speed of communications according to the following equations: Cyclone / TraceLink: (50000000/(2*N+5)) Hz USB Multilink Universal: (1000000/(N+1)) Hz USB ML Universal FX : (25000000/(N+1)) Hz OSJTAG : Fixed Frequency OpenSDA : Fixed Frequency The value n should be between 0 and 31. This shift clock takes effect after the commands in the top of the programming algorithm are executed so that these commands can increase the target frequency and allow a faster shift clock. This clock can't generally exceed a div 4 of the processor bus frequency.
Delay After Reset	Specifies a delay after the programmer resets the target that we check to see if the part has properly gone into background debug mode. This is useful if the target has a reset driver which hold the MCU in reset after the programmer releases the reset line. The n value is a delay in milliseconds.
Always Mass Erase on Connect	Performs a mass erase of the internal flash immediately upon connection. This is used to recover a device which was incorrectly programmed with junk data in its reset vector.
Use SWD Reduced Pin Protocol	Uses the 2-pin serial wire debug (SWD) protocol instead of JTAG. SWD requires 2 fewer pins than JTAG. **NOTE:** For Kinetis L-, E-, and M-Series, SWD protocol must be used.
Trace Max Buffer Size	Only applicable for the P&E TraceLink interface. Configures the trace buffer capacity of the TraceLink. Smaller sizes result in faster trace upload times, whereas larger sizes allow the user to view more trace information.

To change P&E Connections settings, perform these steps:

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

NOTE

It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

   The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection of which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the Freescale Eclipse Extensions Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E ARM Multilink/Multilink Universal/Cyclone /OSJTAG. The P&E connections settings will appear below.
8. Click Refresh to scan valid interface and port.

   Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

NOTE

The port displayed may vary depending on the interface. For example, if you select interface as Cyclone - Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Hardware Interface Power Control (Voltage --> Power -Out Jack) group.

NOTE

This group will be enabled for the Tracelink and USB Multilink Universal FX interfaces only. For USB Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

12. Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
13. Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.
14. Select a voltage value from the Regulator Output Voltage drop-down list. This adjusts the output voltage that powers the hardware target.

CAUTION

An improper voltage setting can damage the board.

15. Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
16. Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
17. Select a BDM speed using the drop-down box in the BDM Debug Shift Frequency.

NOTE

If you select a fast BDM speed, there may be scenarios where you may have difficulty communicating with your target device. You may want to experimentally select a BDM speed that is fast and yet have a good communication with your target.

18. Click on the Delay after Reset checkbox and enter the desired delay (in milliseconds) in the text box.

    This option specifies the time interval to wait between resetting and communicating the target device.

19. Click the OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.
20. Click on the Close button to close the Debug Configuration dialog box.

14.1.3.1.1 P&E Hardware Interface Connection-Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB MultiLink Universal [FX]
- P&E Cyclone Serial P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- P&E TraceLink USB
- P&E TraceLink Ethernet
- Open Source JTAG
- OpenSDA

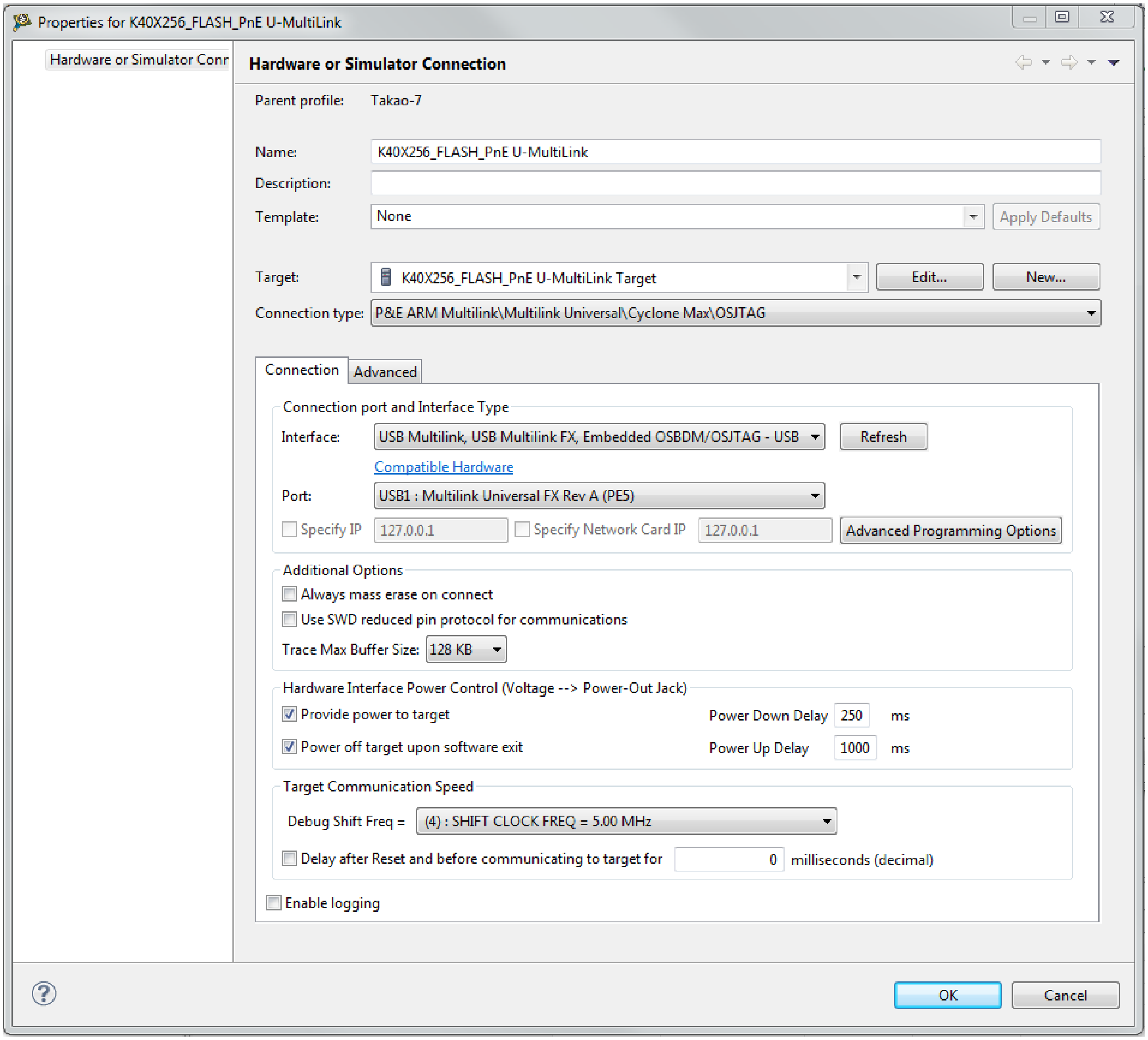
14.1.3.1.1.1 P&E USB MultiLink Universal [FX]

The P&E Kinetis setting permits a connection to USB Multilink devices, which include the USB Multilink Universal, and the USB Multilink Universal FX. P&E USB MultiLink Universal [FX] mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

14.1.3.1.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM Multilink/Multilink Universal/Cyclone / OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

☑ Power off target upon software exit

Power Down Delay 250 ms

Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 14-5. P&E's Launch Configuration Dialog Box

To use P&E’s USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

14.1.3.1.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

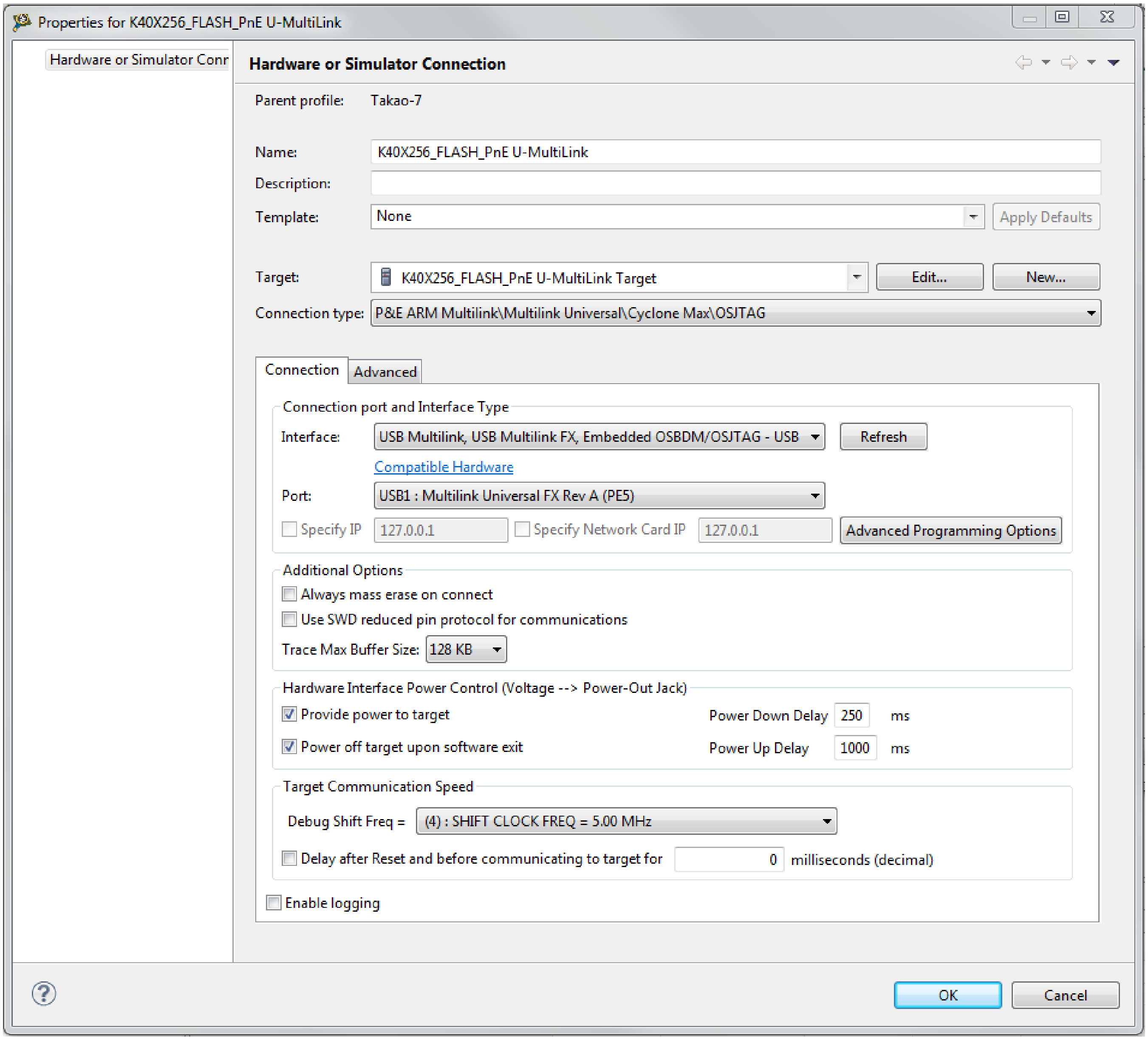
NOTE

> Use of SWD with the Cyclone requires the JTAG SWD Adapter. This can be found at www.pemicro.com.

14.1.3.1.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM V234 Multilink/Multilink Universal/Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None [Apply Defaults]

Target: K40X256_FLASH_PnE U-MultiLink Target [Edit...] [New...]

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB [Refresh]

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1 [Advanced Programming Options]

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

[OK] [Cancel]

Figure 14-6. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone – Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

14.1.3.1.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

NOTE

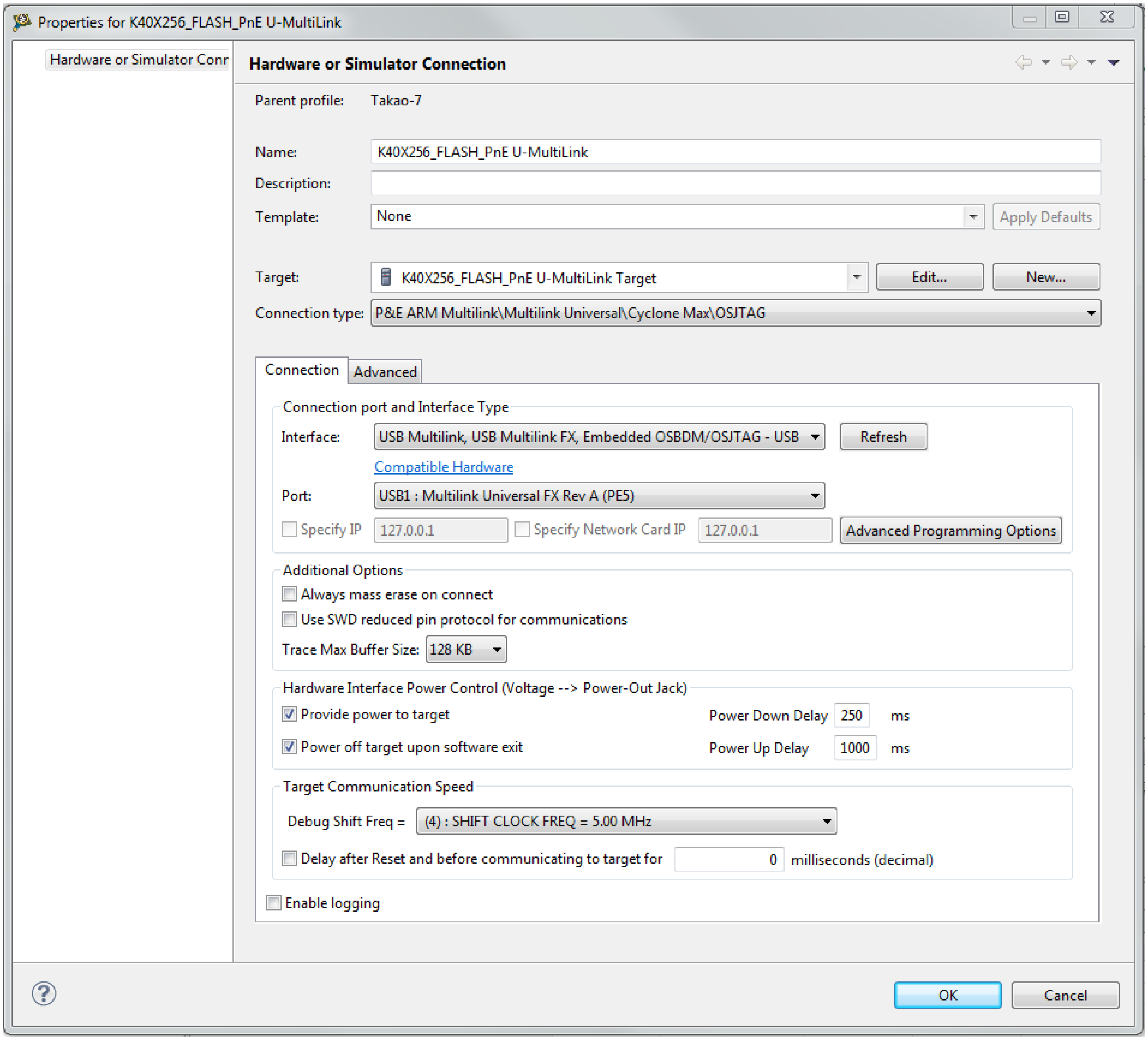
Use of SWD with the Cyclone MAX requires the JTAG SWD Adapter. This can be found at www.pemicro.com

.

14.1.3.1.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM Multilink/Multilink Universal/Cyclone Max/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Apply Defaults

Target: K40X256_FLASH_PnE U-MultiLink Target

Edit... New...

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Refresh

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Advanced Programming Options

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

Power Down Delay 250 ms

☑ Power off target upon software exit

Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

OK Cancel

Figure 14-7. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

14.1.3.1.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

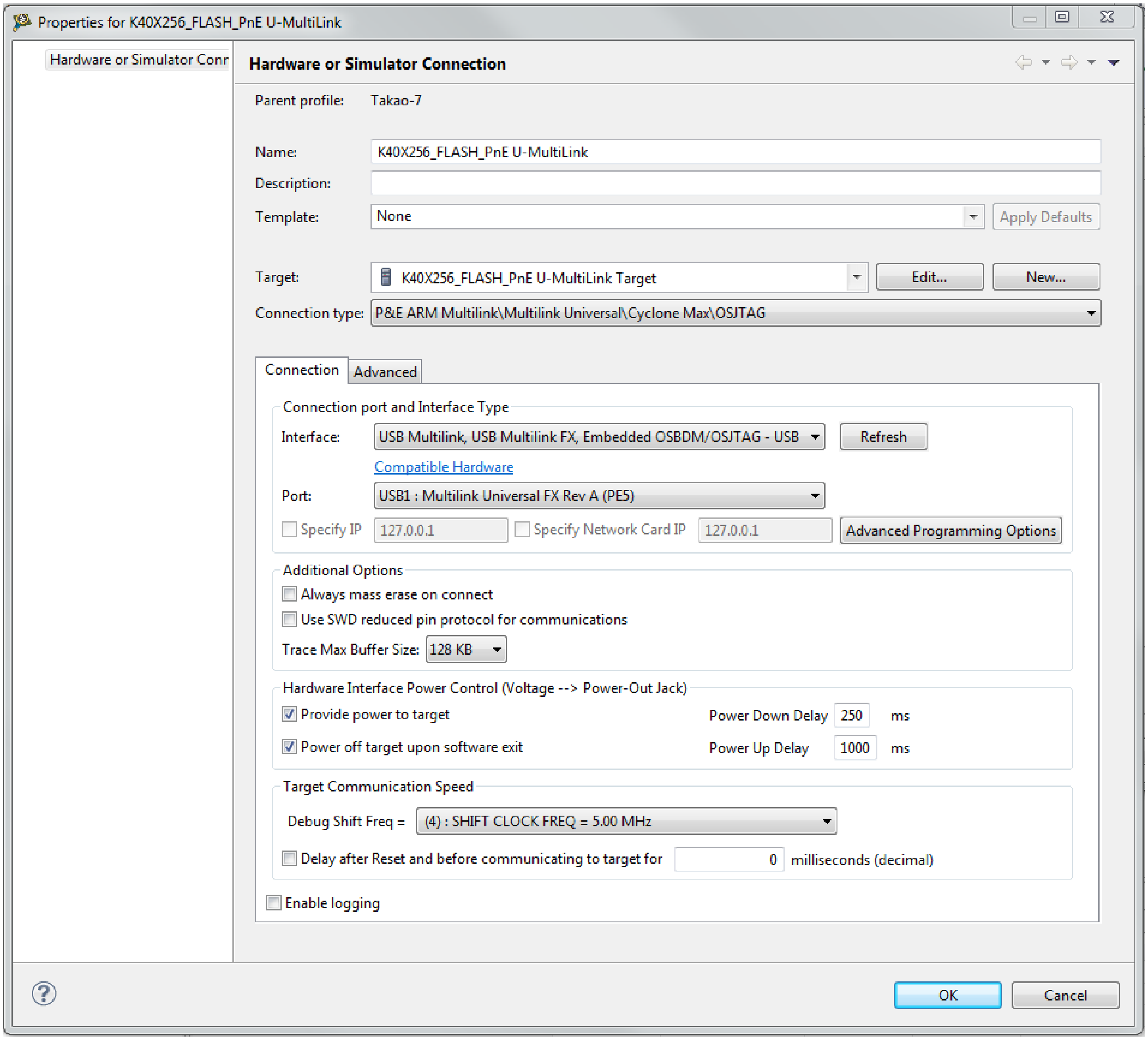
NOTE

Use of SWD with the Cyclone MAX requires the JTAG SWD Adapter. This can be found at www.pemicro.com.

14.1.3.1.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM V234 Multilink/Multilink Universal/Cyclone Max/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 14-8. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone - Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 link to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

14.1.3.1.1.5 P&E TraceLink USB

The P&E TraceLink USB Connection setting permits a connection to TraceLink USB devices. P&E TraceLink USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. P&E's TraceLink can conveniently support all Freescale architectures found in the current CodeWarrior 10 version; however, it can only capture the external trace signals for the ColdFire V2/3/4 and Kinetis ARM architectures.

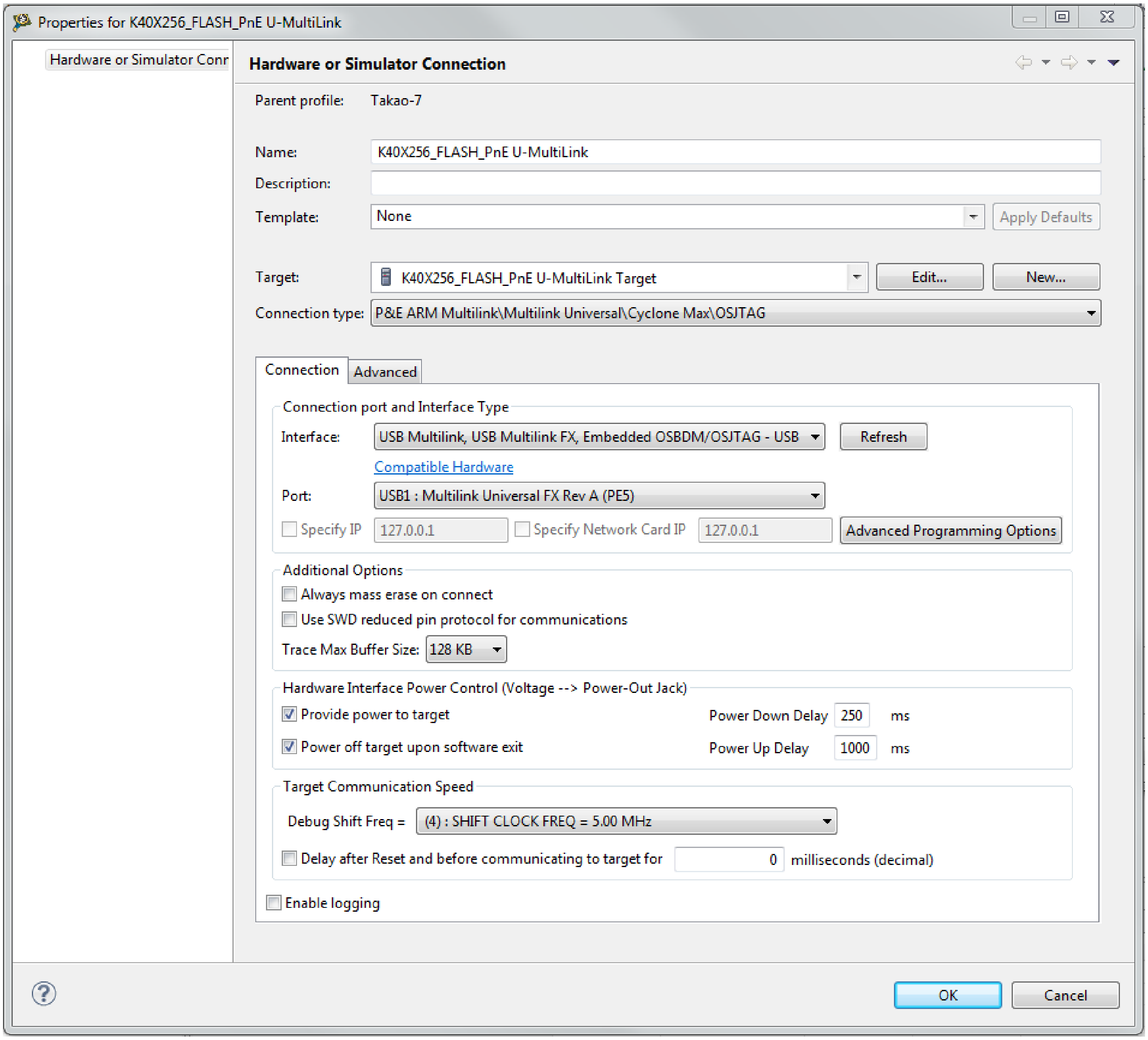
NOTE

> For external trace captures, the P&E TraceLink supports ETM trace. The Kinetis chip must have the Trace_CLKOUT and Trace_D[3:0] pins to support ETM trace. Chips without these pins can still be debugged by the TraceLink within CodeWarrior.

14.1.3.1.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM Multilink/Multilink Universal/Cyclone Max/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 14-9. P&E's Launch Configuration Dialog Box

To use P&E's TraceLink USB, first connect your hardware interface to your computer, and then set the interface to TraceLink - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

14.1.3.1.1.6 P&E TraceLink Ethernet

The P&E TraceLink Ethernet Connection setting permits a connection to TraceLink Ethernet devices. P&E TraceLink Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. P&E's TraceLink can conveniently support all Freescale architectures found in the current CodeWarrior 10 version; however, it can only capture the external trace signals for the ColdFire V2/3/4 and Kinetis ARM architectures.

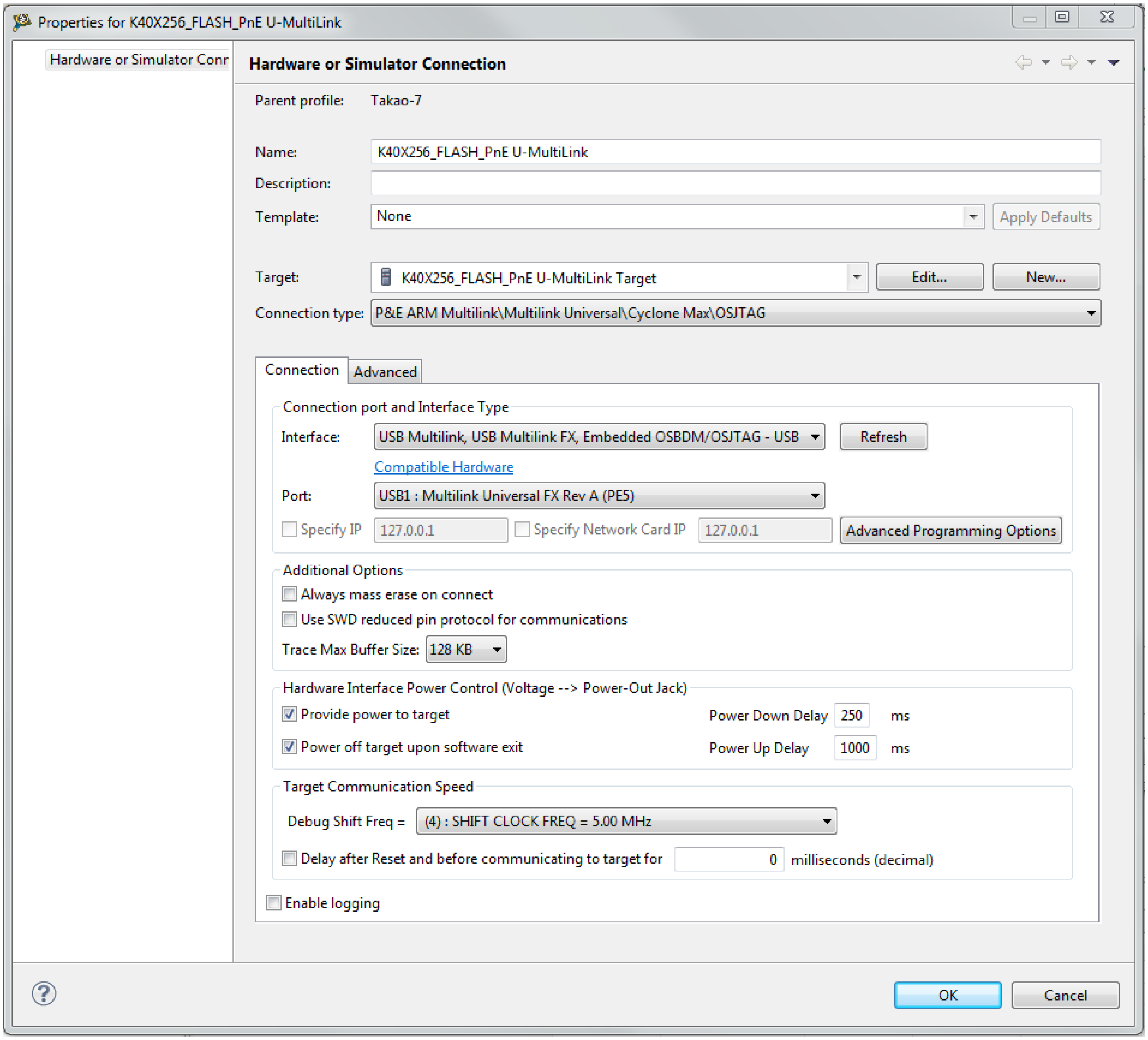
NOTE

> For external trace captures, the P&E TraceLink supports ETM trace. The Kinetis chip must have the Trace_CLKOUT and Trace_D[3:0] pins to support ETM trace. Chips without these pins can still be debugged by the TraceLink within CodeWarrior.

14.1.3.1.1.6.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM Multilink/Multilink Universal/Cyclone Max/ OSJTAG so that the Connection tab will populate P&E’s hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conn

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 14-10. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG/OpenSDA – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

14.1.3.1.1.7 Open Source JTAG

Freescale supplies certain development boards with an integrated debug circuit based on Open Source JTAG. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink Universal. The development board also derives its power from the USB Bus.

The Open Source JTAG circuit design (OSJTAG-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

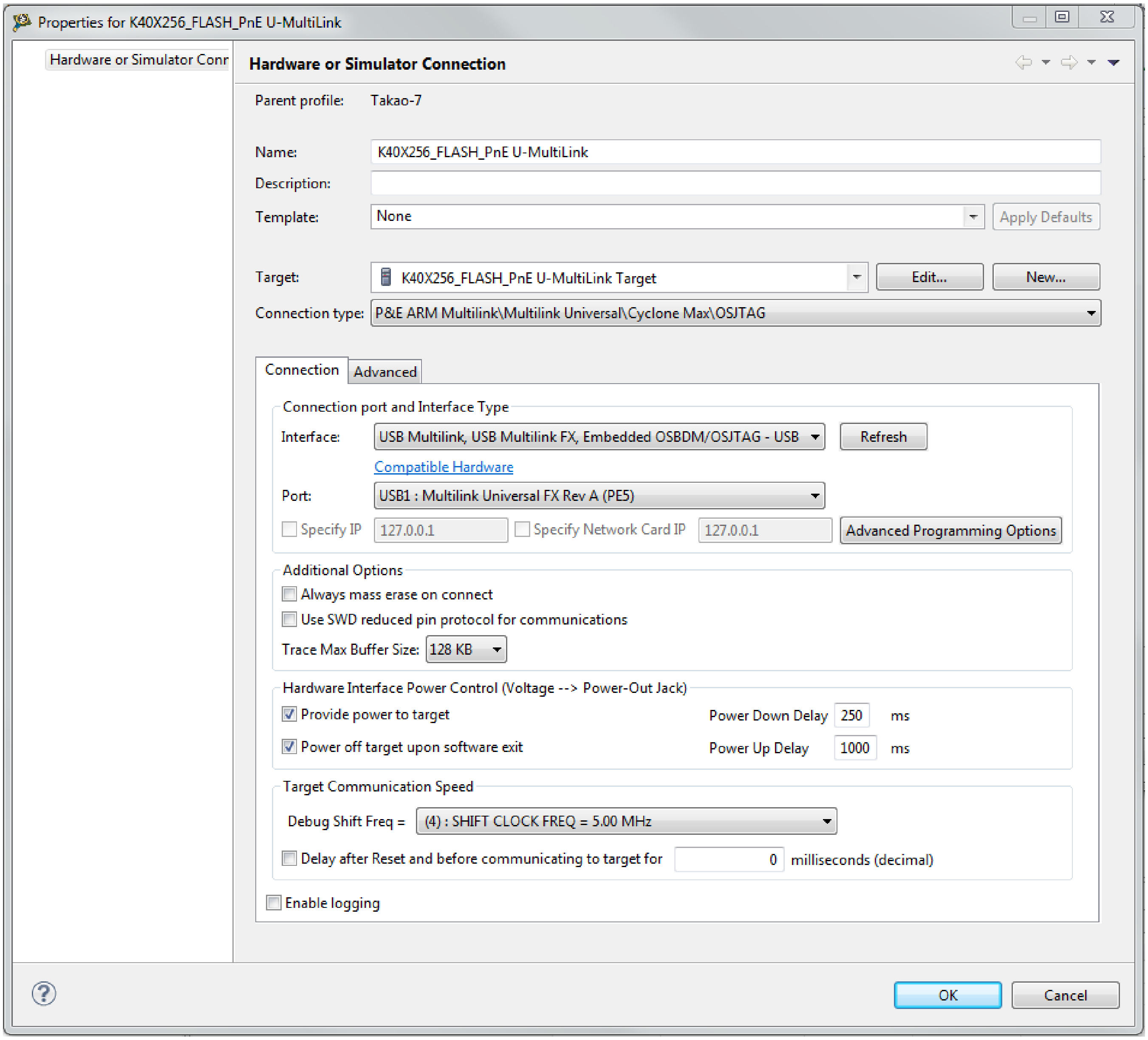
Integration with CodeWarrior is handled via the "Open Source JTAG" connection. P&E has integrated the Open Source JTAG support into the same connection that supports both the USB ColdFire Multilink and the Cyclone . All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSJTAG (albeit at a lower data rate).

The Open Source JTAG Connection setting permits a connection to Open Source JTAG devices. Open Source JTAG mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

14.1.3.1.1.7.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM Multilink/Multilink Universal/Cyclone / OSJTAG so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

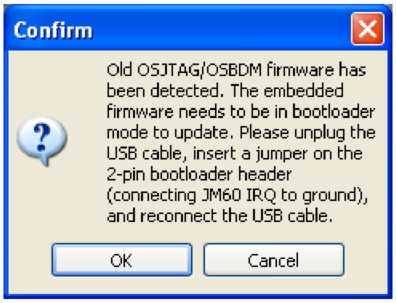
□ Enable logging

Figure 14-11. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG/OpenSDA – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

14.1.3.1.1.7.2 OSJTAG Firmware Update

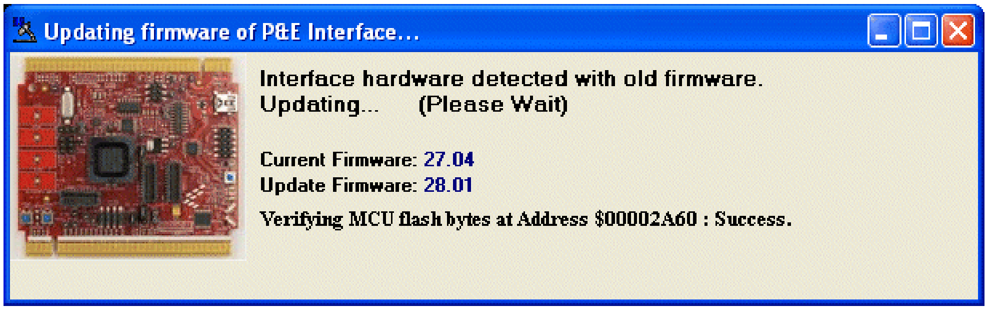
All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSJTAG hardware interfaces. Whenever an OSJTAG-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSJTAG firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.



Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 14-12. Old OSJTAG Firmware Detected

To update the firmware, the OSJTAG device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSJTAG-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSJTAG device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSJTAG device and click OK. If done correctly, the automatic firmware update will occur.

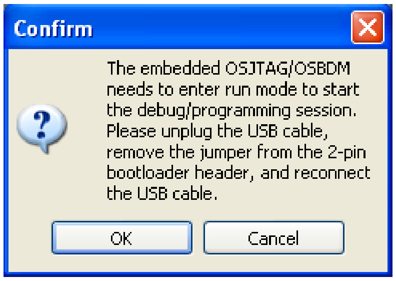


Interface hardware detected with old firmware.
Updating... (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01
Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 14-13. OSJTAG Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSJTAG device must exit Bootloader mode and enter into Run mode.



The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

Figure 14-14. Start OSJTAG Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSJTAG device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSJTAG firmware. If for any reason you experience difficulty performing OSJTAG firmware update, visit `www.pemicro.com/osbdm` and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

14.1.3.1.1.8 OpenSDA

Freescale supplies certain development boards with an integrated debug circuit based on OpenSDA. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink Universal. The development board also derives its power from the USB Bus.

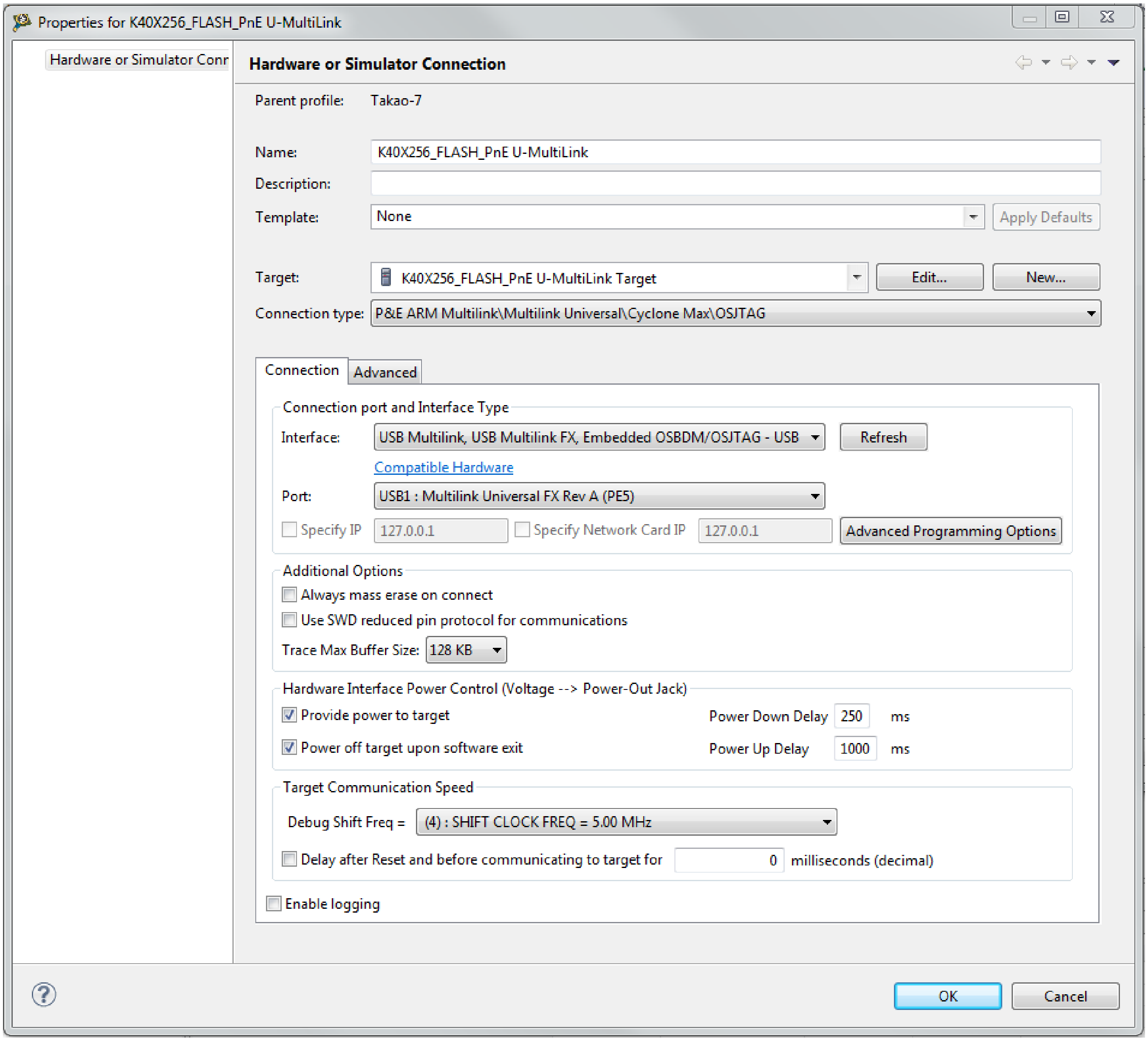
The OpenSDA circuit design is incorporated into many of Freescale's tower cards. The main processor is pre-programmed with a Bootloader. The Bootloader can be used to load P&E's OpenSDA Debug Application, which is shipped with the board. It can also be downloaded from http://www.pemicro.com/opensda.

Integration with CodeWarrior is handled via the "OpenSDA" connection. The OpenSDA Connection setting permits a connection to OpenSDA devices. OpenSDA mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

14.1.3.1.1.8.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E ARM Multilink/Multilink Universal/Cyclone / OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in .



Properties for K40X256_FLASH_PnE U-MultiLink

Hardware or Simulator Conn

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K40X256_FLASH_PnE U-MultiLink

Description:

Template: None

Target: K40X256_FLASH_PnE U-MultiLink Target

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

□ Always mass erase on connect

□ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

Target Communication Speed

Debug Shift Freq = (4) : SHIFT CLOCK FREQ = 5.00 MHz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

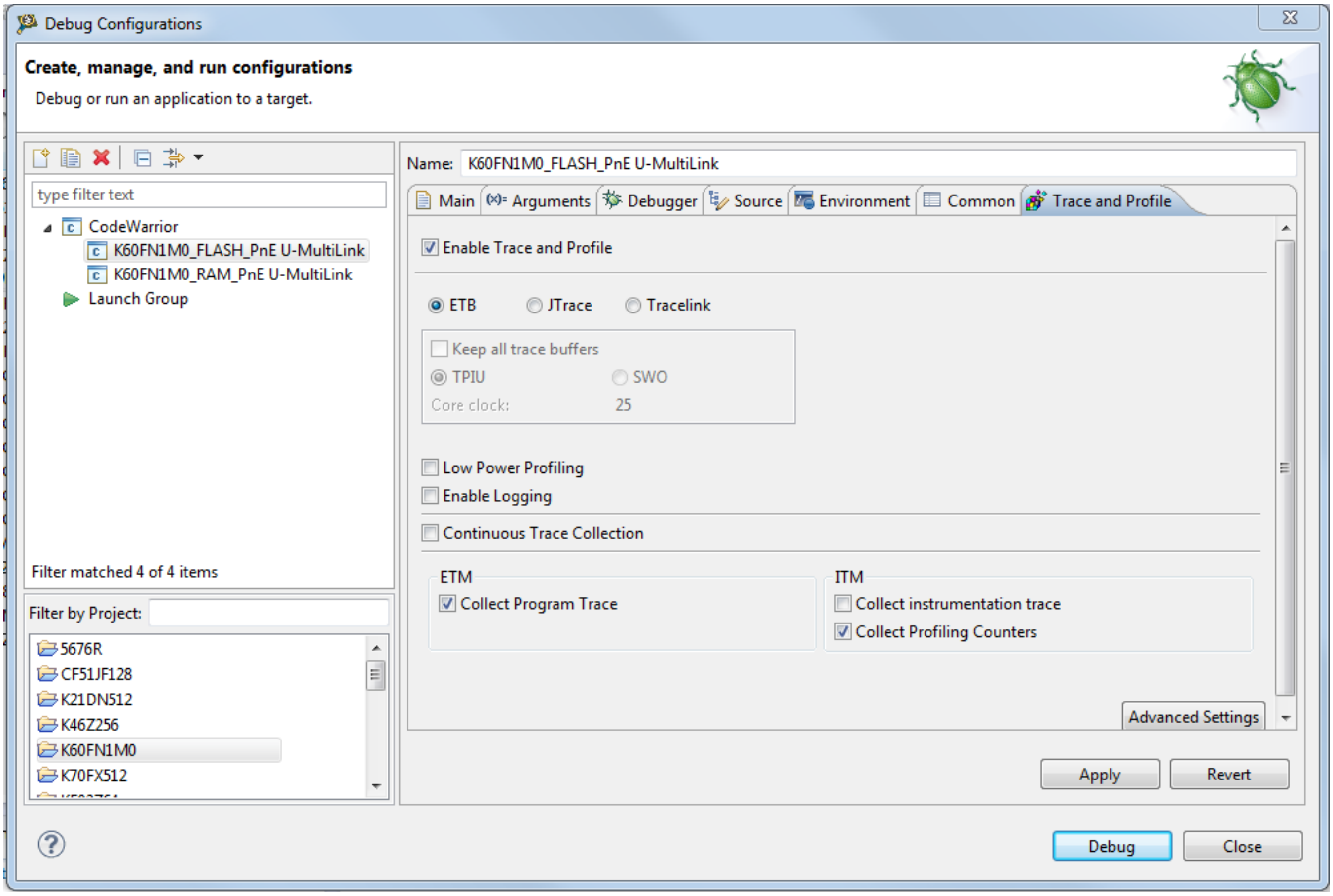
Figure 14-15. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG/OpenSDA – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

14.1.3.1.1.9 Trace and Profile

The P&E TraceLink allows the user to capture real-time external trace information without having to stop or disturb the running application. This allows the user to see the real-time execution of their code by continuously recording the processor events. For external trace captures, the P&E TraceLink supports ETM trace. The Kinetis chip must have Trace_CLKOUT and Trace_D[3:0] pins to support ETM trace. Chips without these pins can still be debugged by the TraceLink within CodeWarrior

1. Right-click on the project and right-click -> Debug as -> Debug Configurations. The Debug Configuration dialog box will appear.
2. In the left column, select the project type for which you would like to set the TraceLink settings.
3. In the right column, select the Trace and Profile tab.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - K60FN1M0_FLASH_PnE U-MultiLink
  - K60FN1M0_RAM_PnE U-MultiLink
- Launch Group

Filter matched 4 of 4 items

Filter by Project:

5676R
CF51JF128
K21DN512
K46Z256
K60FN1M0
K70FX512

Name: K60FN1M0_FLASH_PnE U-MultiLink

☑ Enable Trace and Profile

● ETB ○ JTrace ○ Tracelink

☐ Keep all trace buffers

● TPIU ○ SWO

Core clock: 25

☐ Low Power Profiling

☐ Enable Logging

☐ Continuous Trace Collection

ETM

☑ Collect Program Trace

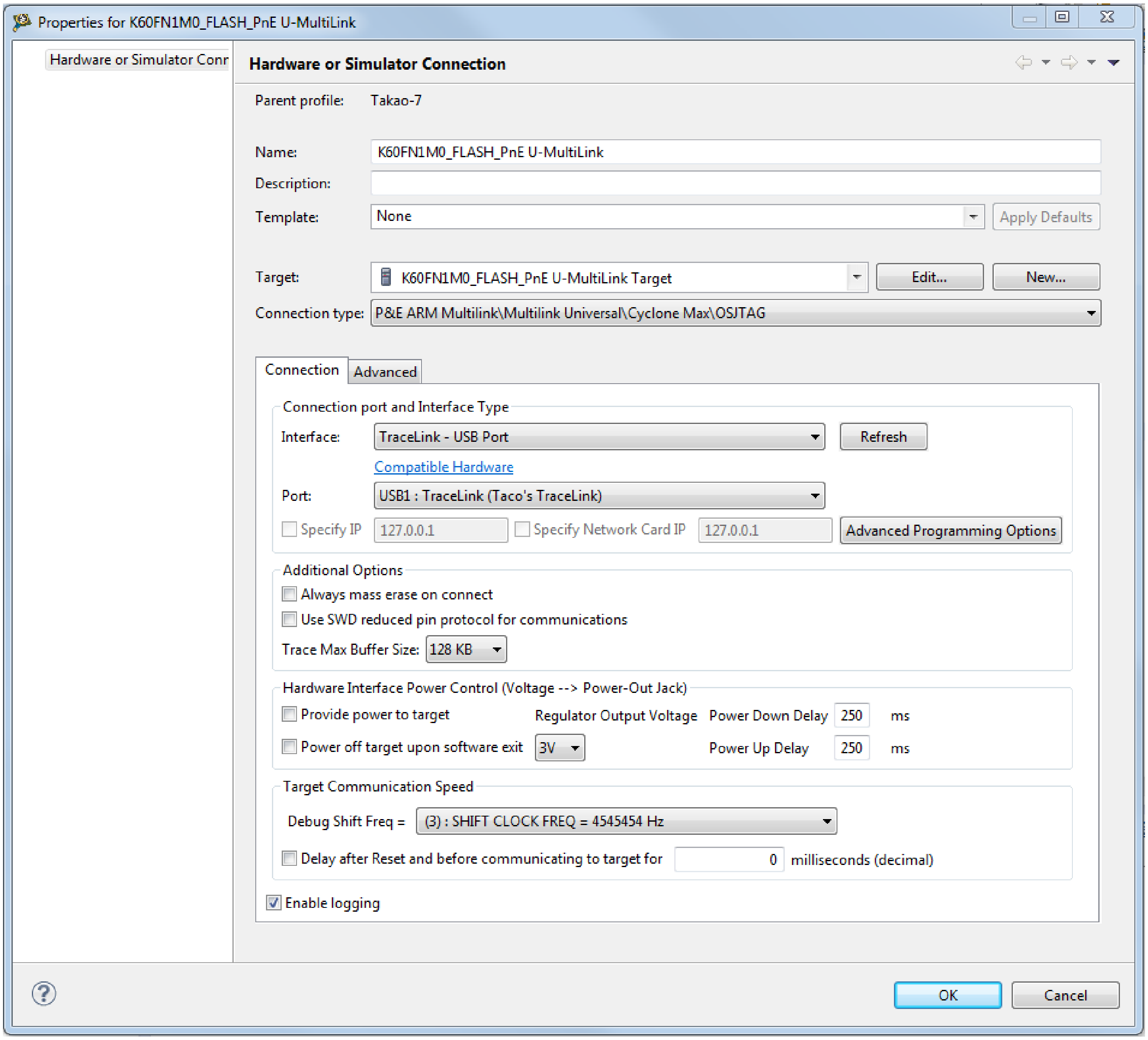
ITM

☐ Collect instrumentation trace

☑ Collect Profiling Counters

Figure 14-16. Trace and Profile Tab

4. Click on the Enable Trace and Profile checkbox.
5. Change the other user settings that fits the users needs.
6. Click on the Main tab and select the correct connection setting from the drop-down box or create a new connection by clicking on the New button.
7. Once the correct connection setting is selected, click on the Edit button. The launch configuration dialog box will appear.
8. When the TraceLink is selected as the interface, the Additional Options will be available. Change the Trace Max Buffer Size as the user sees fit by using the drop-down box. Refer to Figure 14-17.



Properties for K60FN1M0_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: K60FN1M0_FLASH_PnE U-MultiLink

Description:

Template: None [Apply Defaults]

Target: K60FN1M0_FLASH_PnE U-MultiLink Target [Edit...] [New...]

Connection type: P&E ARM Multilink\Multilink Universal\Cyclone Max\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: TraceLink - USB Port [Refresh]

Compatible Hardware

Port: USB1 : TraceLink (Taco's TraceLink)

☐ Specify IP 127.0.0.1 ☐ Specify Network Card IP 127.0.0.1 [Advanced Programming Options]

Additional Options

☐ Always mass erase on connect

☐ Use SWD reduced pin protocol for communications

Trace Max Buffer Size: 128 KB

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☐ Provide power to target Regulator Output Voltage Power Down Delay 250 ms

☐ Power off target upon software exit 3V Power Up Delay 250 ms

Target Communication Speed

Debug Shift Freq = (3) : SHIFT CLOCK FREQ = 4545454 Hz

☐ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

☑ Enable logging

[OK] [Cancel]

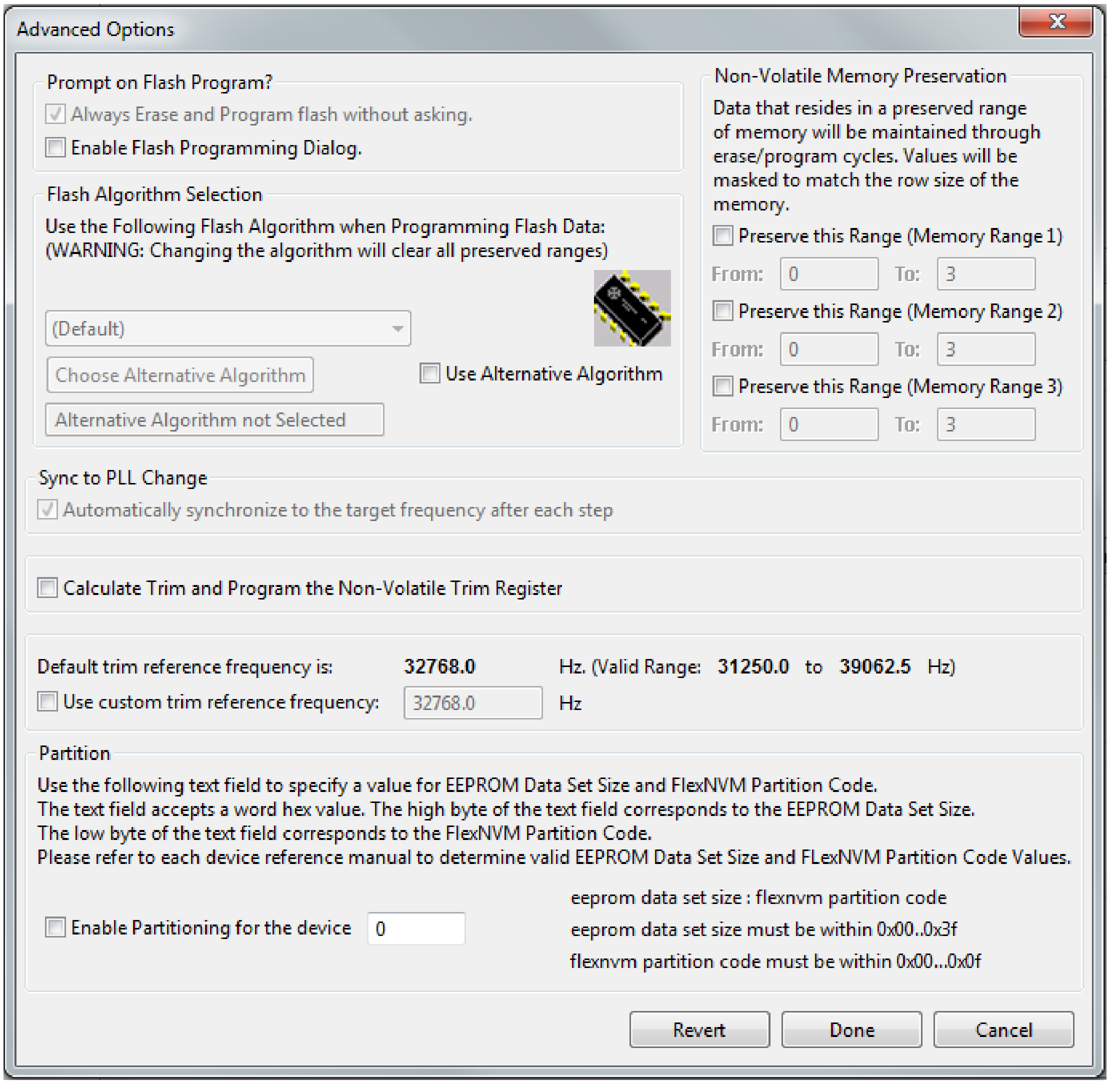
Figure 14-17. Debugger Settings

NOTE

Kinetis L-, E-, and M-Series devices can be debugged by the TraceLink, however Trace and Profile is currently not supported. Kinetis K-Series devices are fully supported.

14.1.3.1.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

□ Use Alternative Algorithm

Alternative Algorithm not Selected

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 0 To: 3

□ Preserve this Range (Memory Range 2)

From: 0 To: 3

□ Preserve this Range (Memory Range 3)

From: 0 To: 3

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

□ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: 32768.0 Hz. (Valid Range: 31250.0 to 39062.5 Hz)

□ Use custom trim reference frequency: 32768.0 Hz

Partition

Use the following text field to specify a value for EEPROM Data Set Size and FlexNVM Partition Code.
The text field accepts a word hex value. The high byte of the text field corresponds to the EEPROM Data Set Size.
The low byte of the text field corresponds to the FlexNVM Partition Code.
Please refer to each device reference manual to determine valid EEPROM Data Set Size and FLexNVM Partition Code Values.

□ Enable Partitioning for the device 0

eeprom data set size : flexnvm partition code
eeprom data set size must be within 0x00..0x3f
flexnvm partition code must be within 0x00...0x0f

Figure 14-18. Advanced Options Dialog Box

14.1.3.1.2.1 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

14.1.3.1.2.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase, and reprogrammed immediately afterwards, thereby preserving the data in these ranges. Any attempt to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

14.1.3.1.2.3 Calculate and Program Non-Volatile Trim

The checkbox gives you the option of trimming device to default center frequency. If this checkbox is selected, a calculated trim frequency will be programmed to a dedicated non-volatile memory location during the next debugging session.

14.1.3.1.2.4 Custom Trim

When the checkbox is checked, you have the ability to input a custom center frequency within an allowed range for a given device. A trim value based on this frequency will be calculated and programming into dedicated non-volatile memory location during the next debug session.

14.1.3.1.2.5 Alternative Algorithm Functionality

Once you create a project for a specific Kinetis microprocessor, the debugger specifies a default algorithm to use during all Flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the <CW_Install>/MCU/bin/plugins/support/ARM/gdi/P&E directory

However, you can override the default algorithm via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. This feature can be used to select a custom programming algorithm, or select another one of P&E's many programming algorithms for use with a specific project.

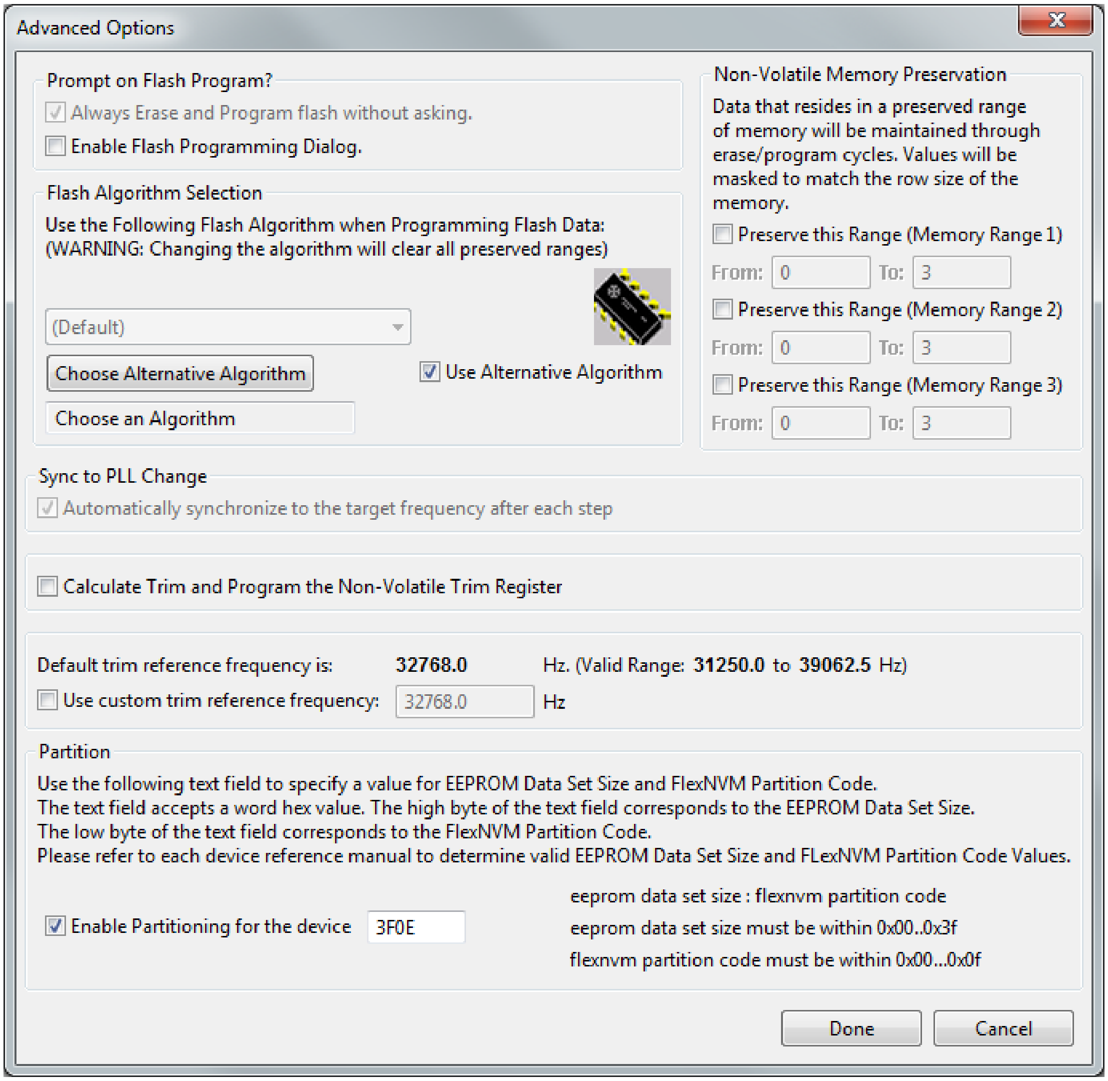
Tip

> Selecting a wrong programming algorithm may damage your device, lead to under/ over programming situations, or simply not program portions of the project file.

Therefore it is recommended to use the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. Check the Use Alternative Algorithm checkbox.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm

☑ Use Alternative Algorithm

Choose an Algorithm

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 0 To: 3

□ Preserve this Range (Memory Range 2)

From: 0 To: 3

□ Preserve this Range (Memory Range 3)

From: 0 To: 3

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

□ Calculate Trim and Program the Non-Volatile Trim Register

Default trim reference frequency is: **32768.0** Hz. (Valid Range: **31250.0** to **39062.5** Hz)

□ Use custom trim reference frequency: 32768.0 Hz

Partition

Use the following text field to specify a value for EEPROM Data Set Size and FlexNVM Partition Code.
The text field accepts a word hex value. The high byte of the text field corresponds to the EEPROM Data Set Size.
The low byte of the text field corresponds to the FlexNVM Partition Code.
Please refer to each device reference manual to determine valid EEPROM Data Set Size and FLexNVM Partition Code Values.

☑ Enable Partitioning for the device 3F0E

eeprom data set size : flexnvm partition code
eeprom data set size must be within 0x00..0x3f
flexnvm partition code must be within 0x00...0x0f

Figure 14-19. Advanced Options - Alternative Algorithm Checkbox

2. Click the Choose Alternative Algorithm button, which lets you browse for an alternative algorithm.
3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the Choose Alternative Algorithm button.

At this point, the current project performs all future Flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

14.1.3.1.2.6 Enable Partitioning

Setting the Enable Partitioning checkbox enables you to partition your Kinetis chip. To setup partitioning, you must provide two bytes in hexadecimal format. The first byte represents the EEPROM Data Set Size, which determines the amount of FlexRAM used in each of the available EEPROM subsystems. This byte must be in the range of 0x00 to 0x3F. The second byte represents the FlexNVM Partition Code, which specifies how to split the FlexNVM block between data flash and EEPROM backup memory supporting EEPROM functions. This must be in the range of 0x00 to 0x0E. Refer to the reference manual specific to the device you are working with for more information on how to specify the EEPROM Data Set Size and FlexNVM Partition Code.

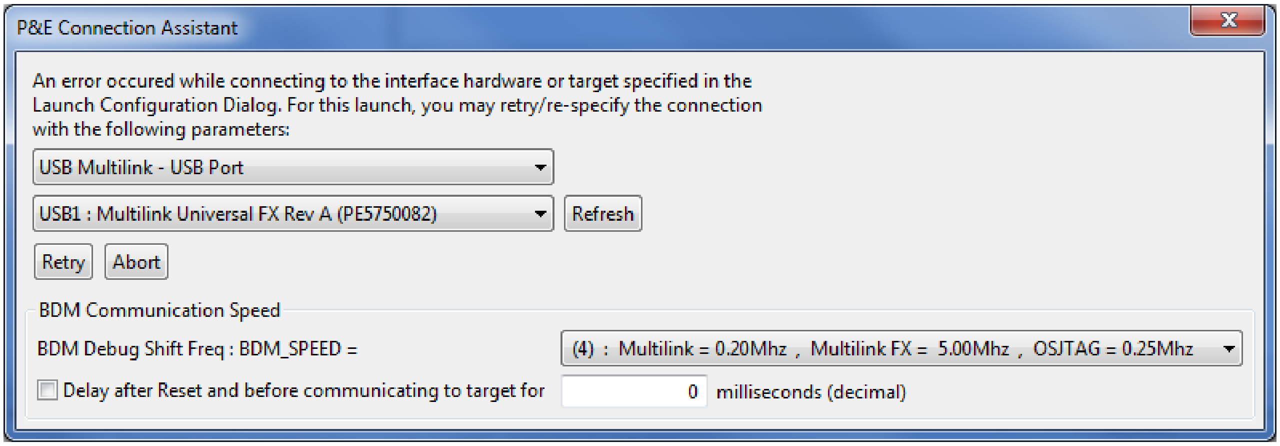
NOTE

This option is only available for Kinetis K-series devices with Flex memory.

14.1.3.2 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug but CodeWarrior cannot connect to the interface hardware specified in the Launch Configuration dialog box. To edit or change your debugger connection:

1. Choose the P&E device that you are using from the first drop-down menu and click Refresh.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the BDM Communication Speed panel to configure the shift frequency and delay.
4. Click the Retry button.



An error occured while connecting to the interface hardware or target specified in the Launch Configuration Dialog. For this launch, you may retry/re-specify the connection with the following parameters:

USB Multilink - USB Port

USB1 : Multilink Universal FX Rev A (PE5750082)

BDM Communication Speed

BDM Debug Shift Freq : BDM_SPEED = (4) : Multilink = 0.20Mhz , Multilink FX = 5.00Mhz , OSJTAG = 0.25Mhz

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

Figure 14-20. Connection Assistant Interface Selected

14.1.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.

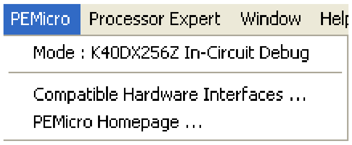

PEMicro
Processor Expert
Window
Help
Mode : K40DX256Z In-Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 14-21. ARM Active Mode Menu

Chapter 15
Connections - DSC Architecture

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the Digital Signal Controller (DSC) target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. The Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

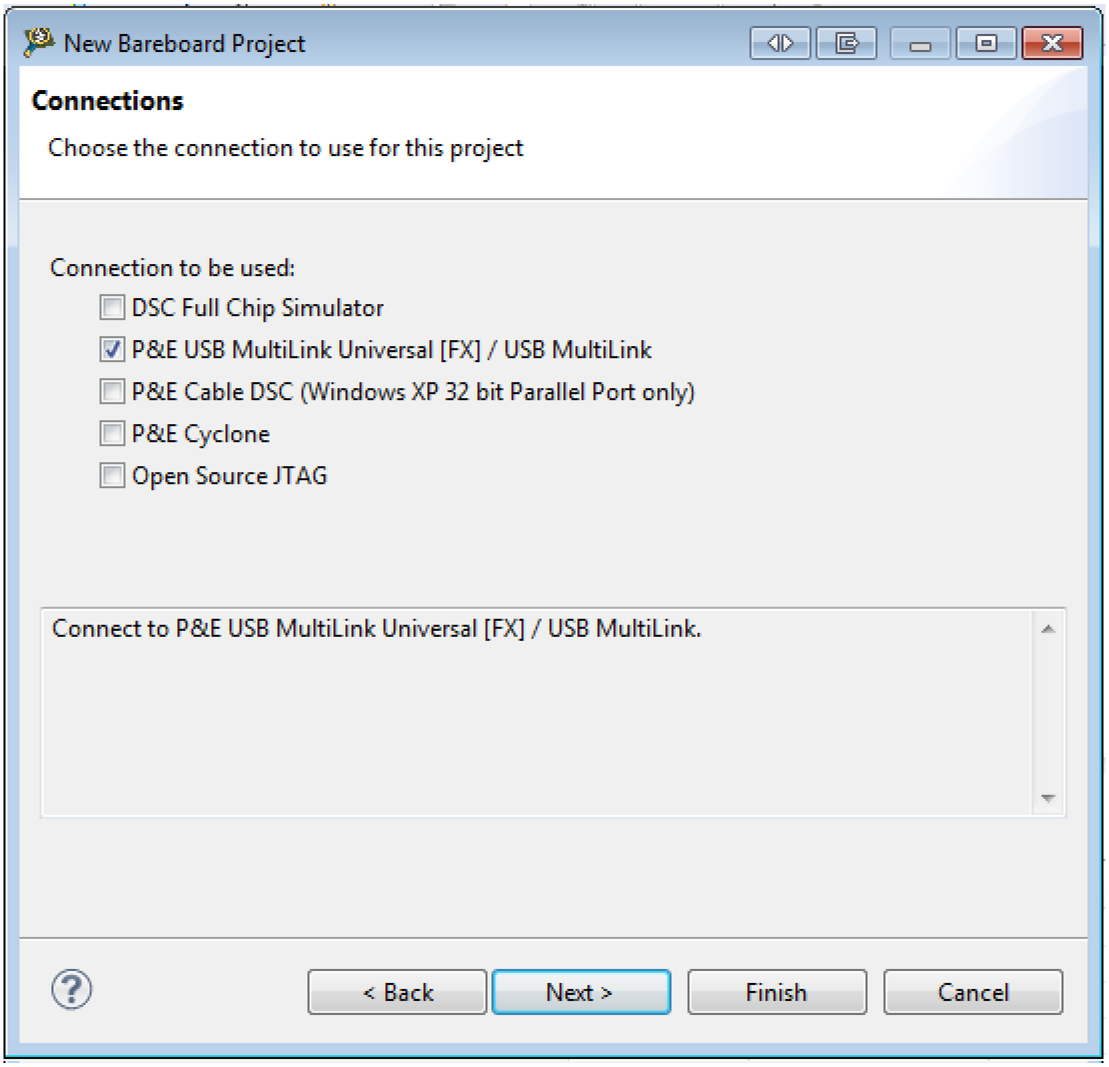
The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with the DSC device family.

15.1 P&E Hardware Interface for DSC

This section describes DSC P&E Connection options. The DSC P&E Connection setting permits a connection to DSC Freescale devices via P&E Multilink, Cyclone (including the Cyclone MAX), and OSJTAG hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

15.1.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.



Connections

Choose the connection to use for this project

Connection to be used:

- □ DSC Full Chip Simulator
- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cable DSC (Windows XP 32 bit Parallel Port only)
- □ P&E Cyclone
- □ Open Source JTAG

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 15-1. New project wizard

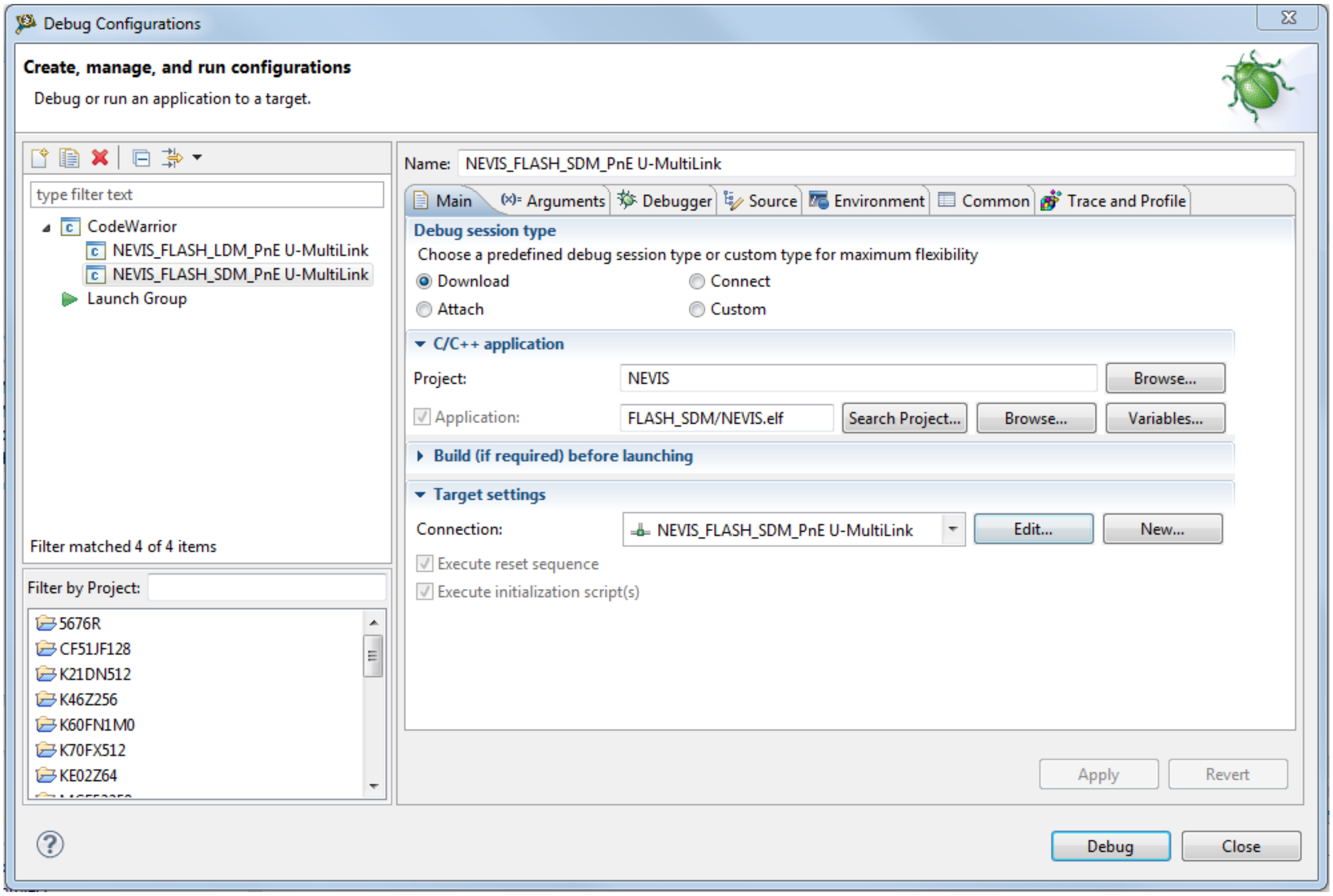
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

15.1.2 Launch Configurations Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project type you would like to set the launch configurations.
3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear.
5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Create, manage, and run configurations

Debug or run an application to a target.

Name: NEVIS_FLASH_SDM_PnE U-MultiLink

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

Project: NEVIS

☑ Application: FLASH_SDM/NEVIS.elf

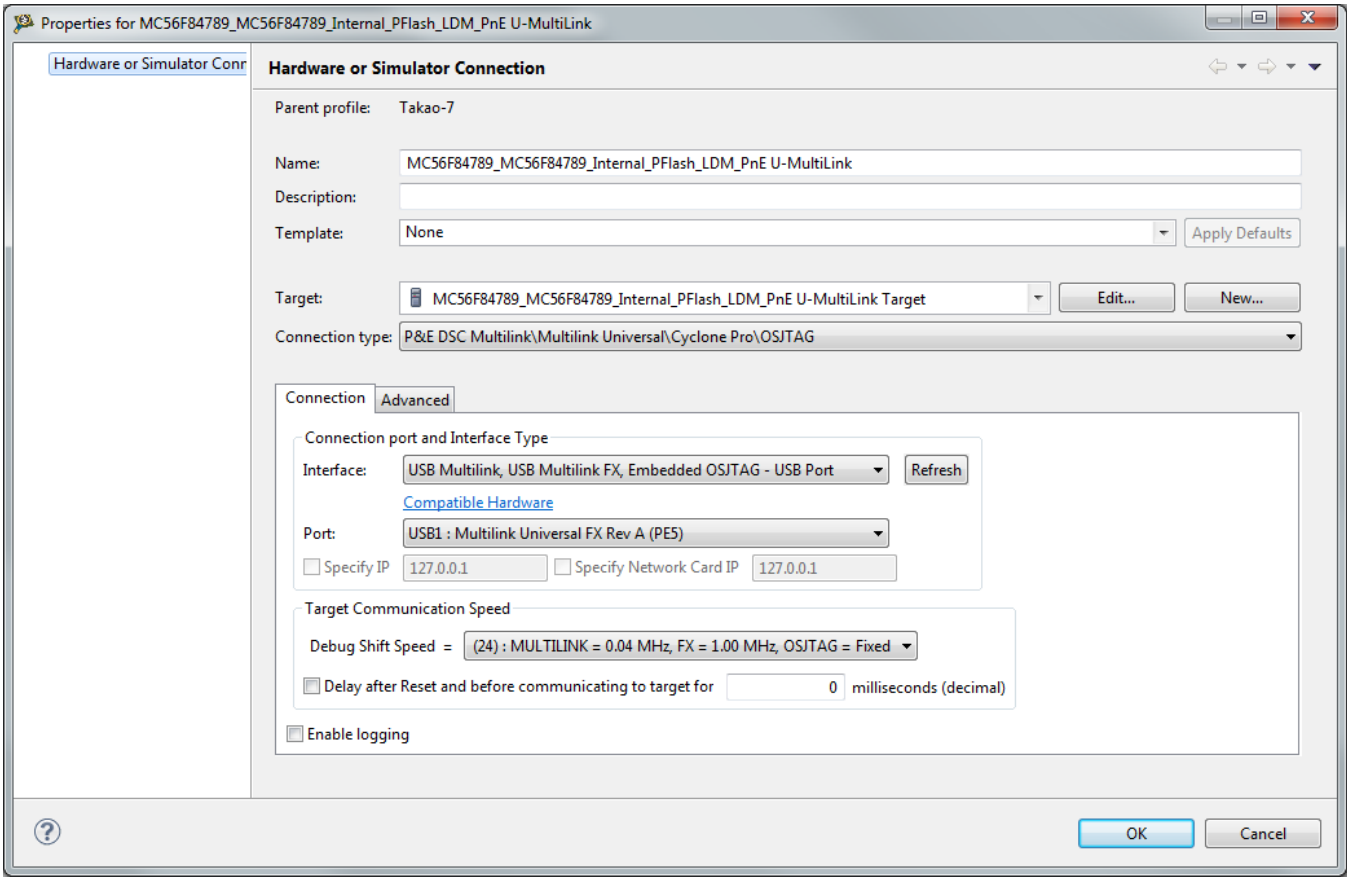
Connection: NEVIS_FLASH_SDM_PnE U-MultiLink

☑ Execute reset sequence

☑ Execute initialization script(s)

Filter matched 4 of 4 items

Figure 15-2. Debug Configuration Dialog Box



Properties for MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Description:

Template: None

Target: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink Target

Connection type: P&E DSC Multilink\Multilink Universal\Cyclone Pro\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSJTAG - USB Port

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Target Communication Speed

Debug Shift Speed = (24) : MULTILINK = 0.04 MHz, FX = 1.00 MHz, OSJTAG = Fixed

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 15-3. Launch Configuration Dialog Box

15.1.3 Connection Options

This topic describes all P&E Multilink/Cyclone /OSJTAG connection options, which are common to all P&E USB Multilink Universal [FX], P&E Cyclone Serial, P&E Cyclone USB, P&E Cyclone Ethernet, P&E Cable DSC, and Open Source JTAG connections.

The options include:

- Changing P&E Connections Settings
- Connection Assistant

15.1.3.1 Changing P&E Connections Settings

All connection settings for P&E hardware interfaces are configured in the Launch Configurations dialog box.

The following table describes the options for this view

Table 15-1. Connection Parameter Options for P&E Multilink/Cyclone /OSJTAG

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB Multilink Universal [FX] - USB Port NOTE: The USB Multilink Universal and the USB Multilink Universal FX can conveniently support all Freescale architectures found in the current CodeWarrior 10 version • Cyclone - Serial Port • Cyclone - USB Port • Cyclone - Ethernet Port • Cable DSC • OSJTAG NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Provide power to target (USB Multilink Universal FX only)	Check this option to have the Cyclone or USB Multilink Universal FX (circuitry) supply power to the hardware target. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Power down delay (USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Debug Shift Freq.	This option lets you to set the debug shift clock speed of P&E's interfaces. This integer value may be used to determine the speed of communications according to the following equations: • Cyclone : (50000000/(2*N+5)) Hz • USB Multilink Universal: (1000000/(N+1)) Hz • USB ML Universal FX : (25000000/(N+1)) Hz • OSJTAG : Fixed Frequency The value n should be between 0 and 31. This shift clock takes effect after the commands in the top of the programming algorithm are executed so that these

Table continues on the next page...

Table 15-1. Connection Parameter Options for P&E Multilink/Cyclone /OSJTAG (continued)

Option	Description
	commands can increase the target frequency and allow a faster shift clock. This clock can't generally exceed a div 4 of the processor bus frequency.
Delay After Reset	Specifies a delay after the programmer resets the target that we check to see if the part has properly gone into background debug mode. This is useful if the target has a reset driver which hold the MCU in reset after the programmer releases the reset line. The n value is a delay in milliseconds.

To change P&E Connections settings, perform these steps:

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

NOTE

It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

   The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection of which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E DSC Multilink/Multilink Universal/Cyclone /OSJTAG. The P&E connections settings will appear below
8. Click Refresh to scan valid interface and port.

   Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

NOTE

The port displayed may vary depending on the interface. For example, if you select interface as Cyclone - Serial Port, the available port option is COM1 : Serial Port 1.

11. Specify settings in the Hardware Interface Power Control (Voltage --> Power -Out Jack) group.

NOTE

This group will be enabled for the USB Multilink Universal FX interface only. For USB Multilink Universal FX interface, use the jumper settings located at JP10 to provide either 3.3V or 5V.

- Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
- Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.
- Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
- Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
- Select a BDM speed using the drop-down box in the BDM Debug Shift Frequency.

NOTE

If you select a fast BDM speed, there may be scenarios where you may have difficulty communicating with your target device. You may want to experimentally select a BDM speed that is fast and yet have a good communication with your target.

- Click on the Delay after Reset checkbox and enter the desired delay (in milliseconds) in the text box.

This option specifies the time interval to wait between resetting and communicating the target device.

- Click the OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.

12. Click on the Close button to close the Debug Configuration dialog box.

15.1.3.1.1 P&E Hardware Interface Connection- Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- P&E Cable DSC
- Open Source JTAG

15.1.3.1.1.1 P&E USB Multilink Universal [FX]

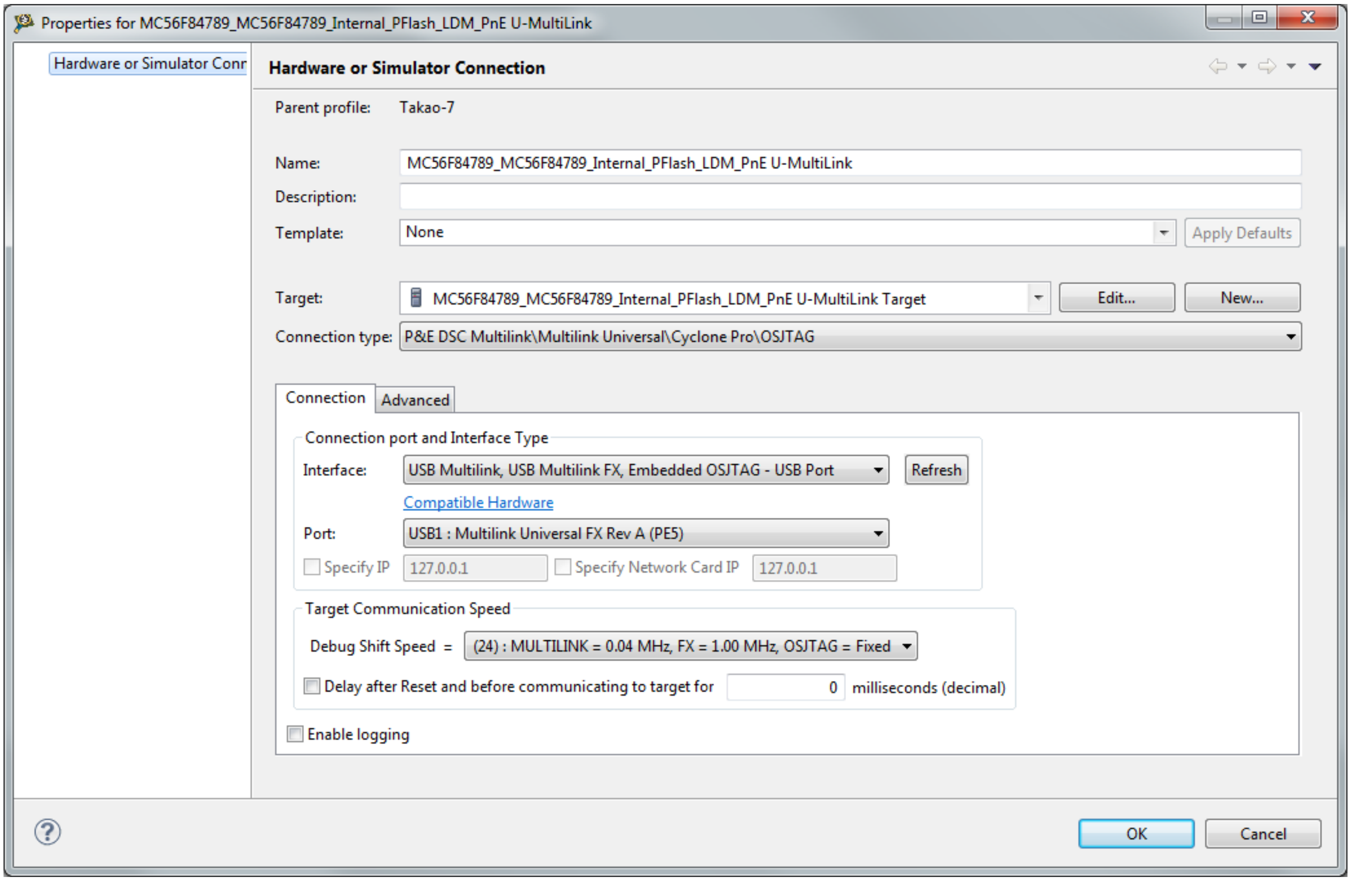
The P&E DSC Connection setting permits a connection to USB Multilink devices, which include the P&E USB Multilink Universal, and the USB Multilink Universal FX. P&E USB Multilink Universal [FX] mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

15.1.3.1.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.

3. Set the connection type to P&E DSC Multilink/Multilink Universal/Cyclone / OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Hardware or Simulator Conn

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Description:

Template: None

Apply Defaults

Target: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink Target

Edit... New...

Connection type: P&E DSC Multilink\Multilink Universal\Cyclone Pro\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSJTAG - USB Port

Refresh

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Target Communication Speed

Debug Shift Speed = (24) : MULTILINK = 0.04 MHz, FX = 1.00 MHz, OSJTAG = Fixed

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

OK Cancel

Figure 15-4. P&E's Launch Configuration Dialog Box

To use P&E's USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

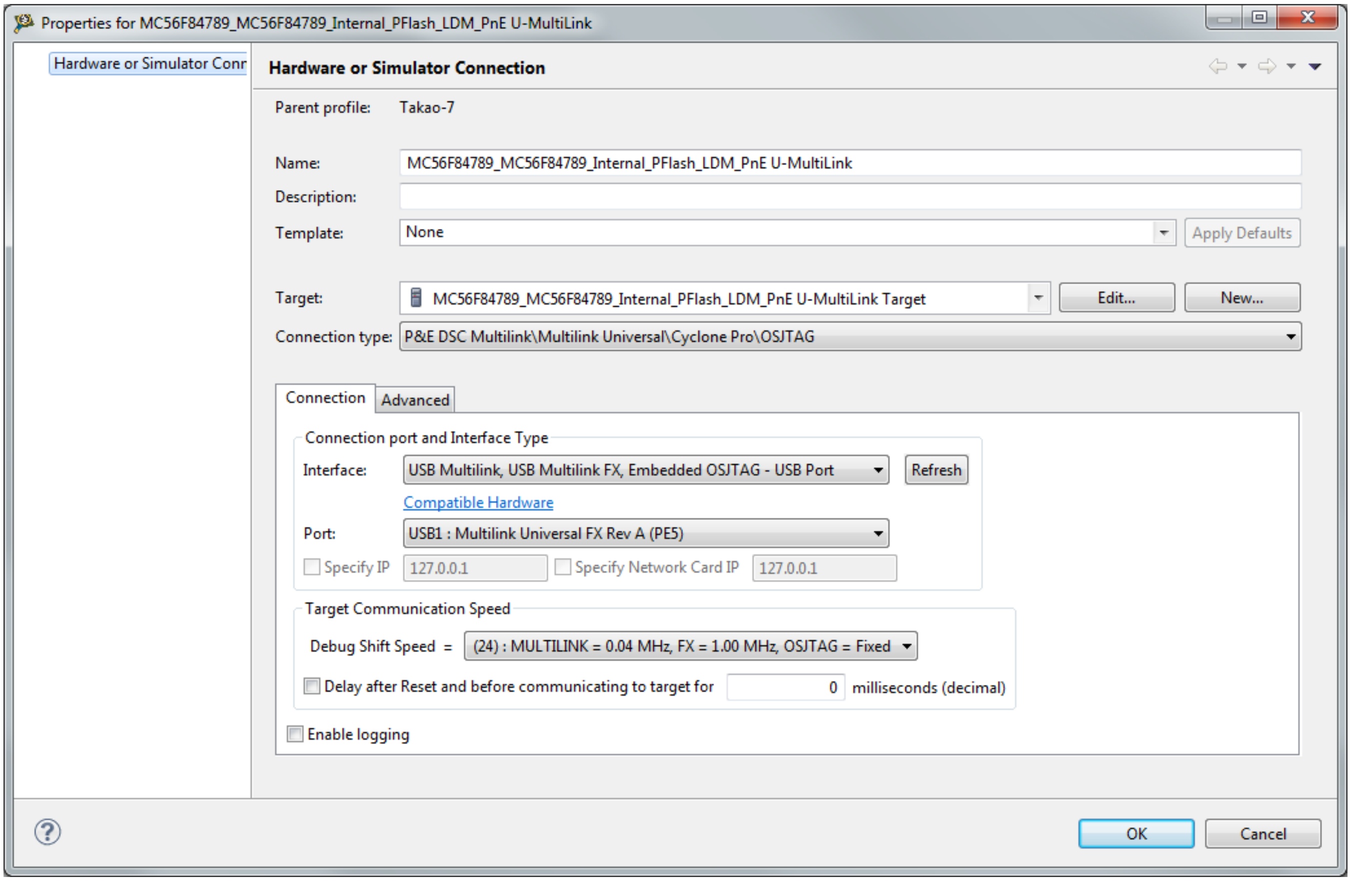
15.1.3.1.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

15.1.3.1.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E DSC Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Description:

Template: None

Target: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink Target

Connection type: P&E DSC Multilink\Multilink Universal\Cyclone Pro\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSJTAG - USB Port

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Target Communication Speed

Debug Shift Speed = (24) : MULTILINK = 0.04 MHz, FX = 1.00 MHz, OSJTAG = Fixed

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 15-5. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone – Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target

.

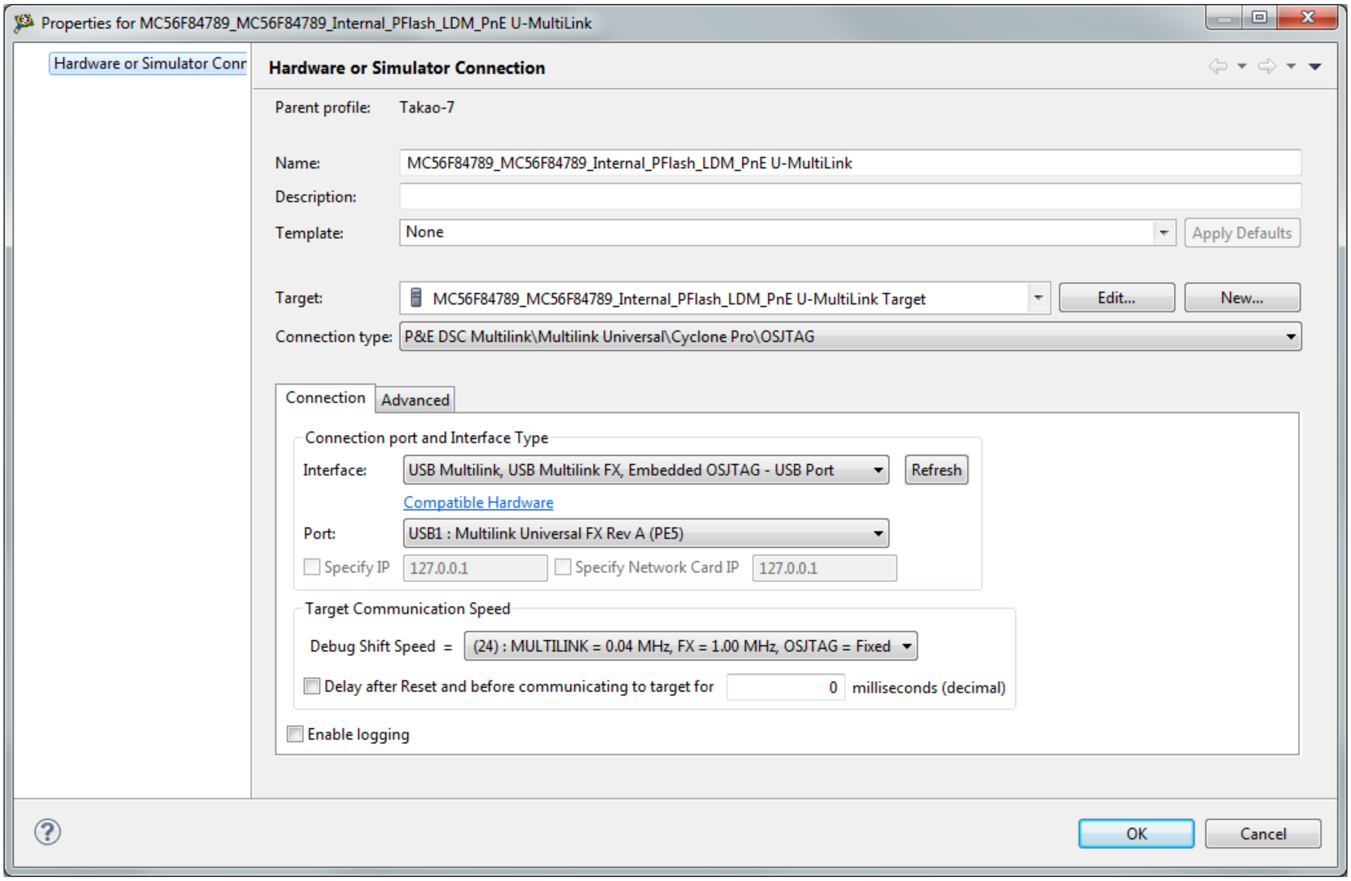
15.1.3.1.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

15.1.3.1.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E DSC Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Description:

Template: None | Apply Defaults

Target: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink Target | Edit... | New...

Connection type: P&E DSC Multilink\Multilink Universal\Cyclone Pro\OSJTAG

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSJTAG - USB Port | Refresh

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Target Communication Speed

Debug Shift Speed = (24) : MULTILINK = 0.04 MHz, FX = 1.00 MHz, OSJTAG = Fixed

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

OK | Cancel

Figure 15-6. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

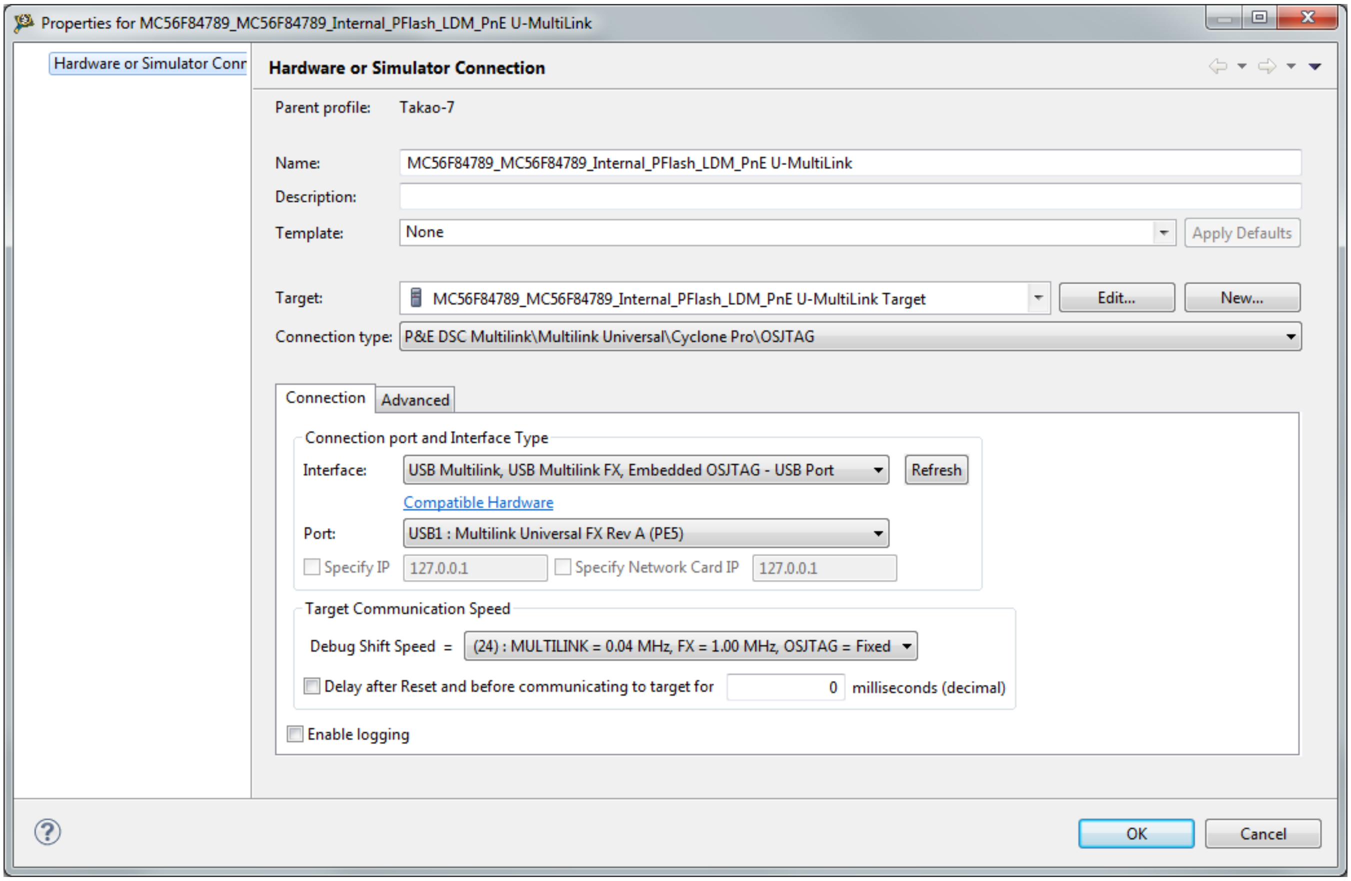
15.1.3.1.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

15.1.3.1.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E DSC Multilink/Multilink Universal/Cyclone Pro/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Hardware or Simulator Conn

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Description:

Template: None

Target: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink Target

Connection type: P&E DSC Multilink\Multilink Universal\Cyclone Pro\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSJTAG - USB Port

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Target Communication Speed

Debug Shift Speed = (24) : MULTILINK = 0.04 MHz, FX = 1.00 MHz, OSJTAG = Fixed

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

□ Enable logging

Figure 15-7. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone – Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

NOTE

The Cyclone MAX does not support the ability to provide power to the target.

15.1.3.1.1.5 P&E Cable DSC

The P&E Cable DSC Connection setting permits a connection to Cable DSC devices. P&E Cable DSC mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

15.1.3.1.1.6 Open Source JTAG

Freescale supplies certain development boards with an integrated debug circuit based on Open Source JTAG. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink Universal [FX]. The development board also derives its power from the USB Bus.

The Open Source JTAG circuit design (OSJTAG-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

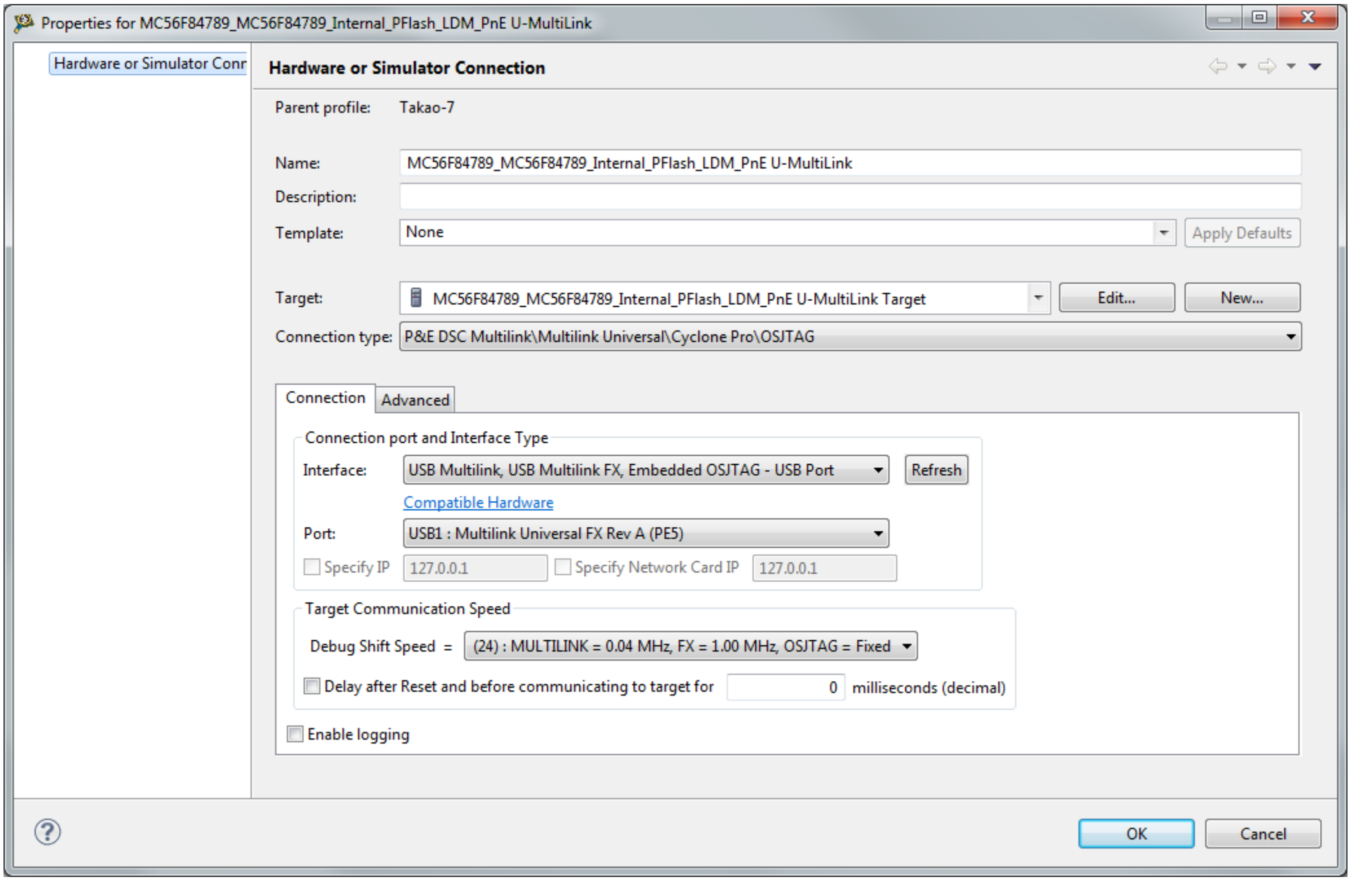
Integration with CodeWarrior is handled via the "P&E Open Source TAG" connection. P&E has integrated the Open Source JTAG support into the same connection that supports both the USB Multilink Universal [FX] and the Cyclone . All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSJTAG (albeit at a lower data rate).

The Open Source JTAG Connection setting permits a connection to Open Source JTAG devices. Open Source JTAG mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

15.1.3.1.1.6.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E DSC V234 Multilink/Multilink Universal/Cyclone MOSJTAG so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink

Description:

Template: None

Target: MC56F84789_MC56F84789_Internal_PFlash_LDM_PnE U-MultiLink Target

Connection type: P&E DSC Multilink\Multilink Universal\Cyclone Pro\OSJTAG

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSJTAG - USB Port

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Target Communication Speed

Debug Shift Speed = (24) : MULTILINK = 0.04 MHz, FX = 1.00 MHz, OSJTAG = Fixed

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

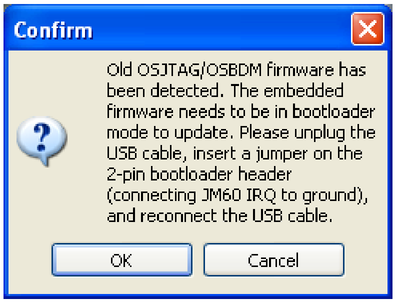
□ Enable logging

Figure 15-8. P&E's Launch Configuration Dialog Box

To use Open Source JTAG, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

15.1.3.1.1.6.2 OSJTAG Firmware Update

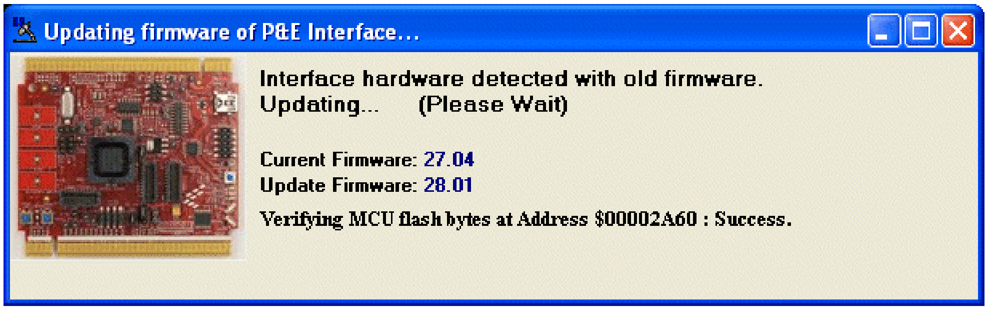
All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSJTAG hardware interfaces. Whenever an OSJTAG-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSJTAG firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.



Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 15-9. Old OSJTAG Firmware Detected

To update the firmware, the OSJTAG device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSJTAG-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSJTAG device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSJTAG device and click on the OK button. If done correctly, the automatic firmware update will occur.



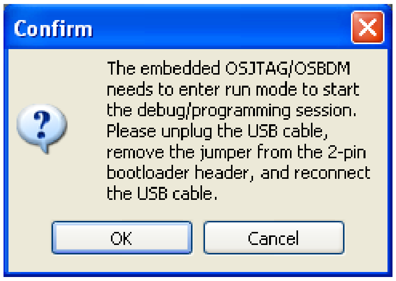
Interface hardware detected with old firmware.
Updating... (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01

Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 15-10. OSJTAG Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSJTAG device must exit Bootloader mode and enter into Run mode.



The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

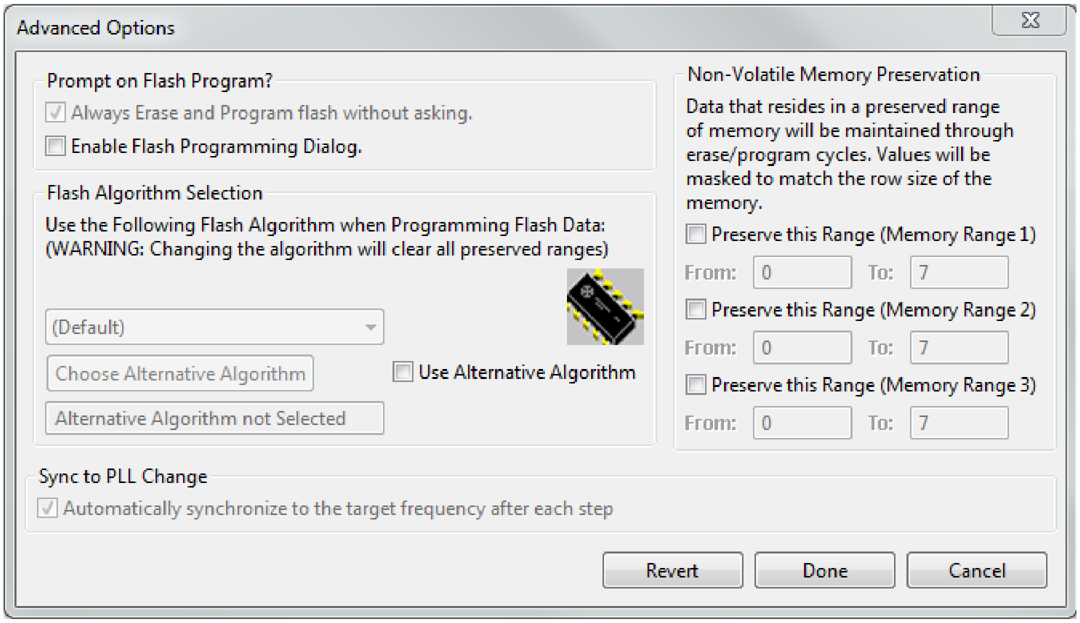
Figure 15-11. Start OSJTAG Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSJTAG device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSJTAG firmware. If for any reason you experience difficulty performing OSJTAG firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

15.1.3.1.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

□ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

□ Use Alternative Algorithm

Alternative Algorithm not Selected

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

□ Preserve this Range (Memory Range 1)

From: 0 To: 7

□ Preserve this Range (Memory Range 2)

From: 0 To: 7

□ Preserve this Range (Memory Range 3)

From: 0 To: 7

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

Figure 15-12. Advanced Options Dialog Box

15.1.3.1.2.1 Alternative Algorithm Functionality

Once you create a project for a specific S12Z microprocessor, the debugger specifies a default algorithm to use during all flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the <CW_Install>/MCU/bin/plugins/support/S12Z/gdi/P&E directory

However, you can override the default algorithm via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. This feature can be used to select a custom programming algorithm or select another one of P&E's many programming algorithms for use with a specific project.

CAUTION

> Selecting the wrong programming algorithm may damage your device, lead to under/over programming situations, or simply not program portions of the project file. Therefore it is recommended to use the default algorithm unless there is a compelling reason to do otherwise.

Use these steps to override the default algorithm:

1. Check the Use Alternative Algorithm checkbox.
2. Click the Choose Alternative Algorithm button, which lets you browse for an alternative algorithm.

3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the Choose Alternative Algorithm button.
4. At this point, the current project will perform all future flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

15.1.3.1.2.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase and reprogrammed immediately afterwards, thereby preserving the data in these ranges. Any attempt to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

15.1.3.1.2.3 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

15.1.3.2 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug but CodeWarrior cannot connect to the interface hardware specified in the Launch Configuration dialog box. To edit or change your debugger connection: 1. Choose the P&E device that you are using from

1. Choose the P&E device that you are using from the first drop-down menu and click Refresh.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the BDM Communication Speed panel to configure the shift frequency and delay.
4. Click the Retry button.

15.1.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.

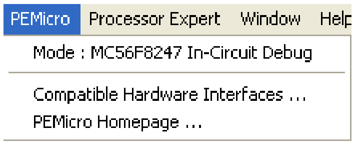

PEMicro
Processor Expert
Window
Help
Mode : MC56F8247 In-Circuit Debug
Compatible Hardware Interfaces ...
PEMicro Homepage ...

Figure 15-13. Additional Connection Menu Options

Chapter 16
Connections - S12Z Architecture

This chapter describes the features and settings of the connections that interface the CodeWarrior debugger with the S12Z target board.

For the IDE to communicate with the target hardware, you must specify several key items: the debugger protocol, a connection type, and any connection parameters. You can enter these items using options in the Launch Configuration panel. Launch Configuration panel can be accessed by clicking on the Edit button located within the Main tab of the Debug Configurations dialog box. These options are:

- The Connection Type option determines what debugger protocol the debugger uses to communicate with the target.
- After you make the option for the connection type, the Connection Settings changes to display configuration options specific for the hardware probe.

The topics in this chapter discuss the features and settings of the connections that interface the CodeWarrior debugger with hardware devices that are part of the S12Z device family.

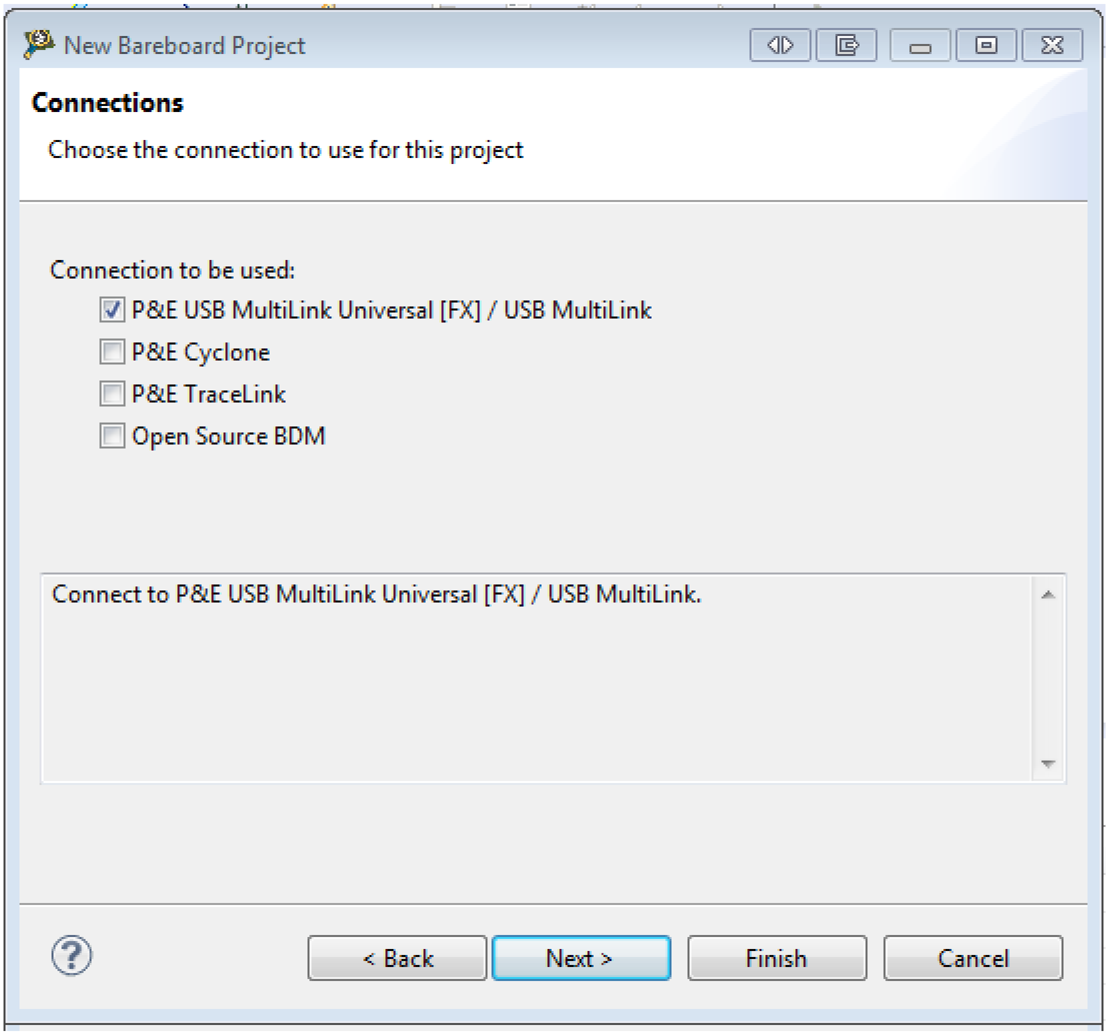
The topics in this chapter are:

16.1 P&E Hardware Interface for S12Z

This section describes the S12Z P&E Connection options. The S12Z Connection setting permits a connection to S12Z Freescale devices via P&E Multilink, Cyclone (including Cyclone PRO), Tracelink, and OSBDM hardware interfaces. This connection mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor.

16.1.1 New Project Wizard

When creating a new project using the New Project Wizard, you will be given the option to select which hardware you will be using to debug your chip. Select the P&E hardware interfaces you want to use by checking the checkboxes.



Connections

Choose the connection to use for this project

Connection to be used:

- ☑ P&E USB MultiLink Universal [FX] / USB MultiLink
- □ P&E Cyclone
- □ P&E TraceLink
- □ Open Source BDM

Connect to P&E USB MultiLink Universal [FX] / USB MultiLink.

Figure 16-1. New project wizard

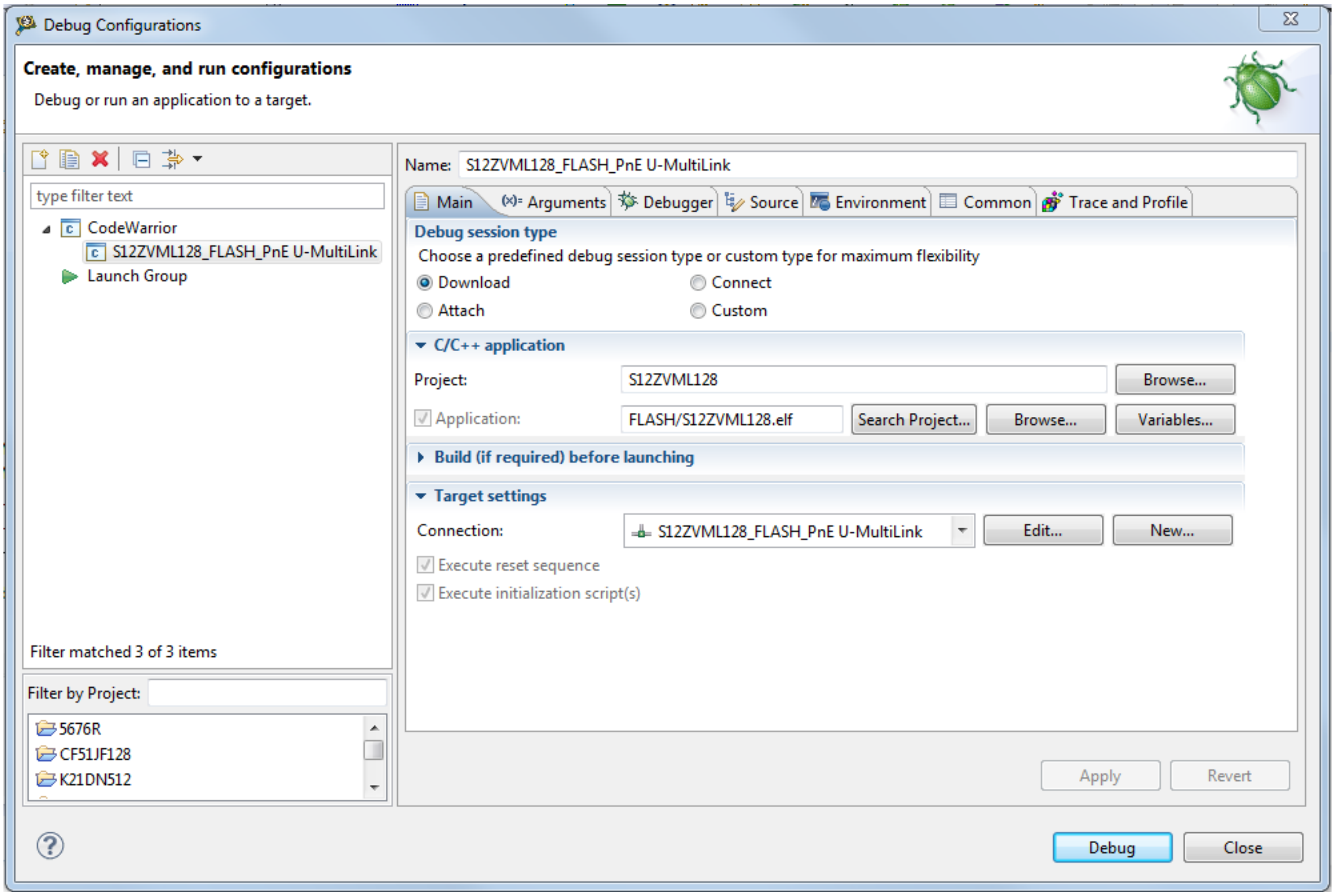
NOTE

Once the project is created, new connections will be created with the P&E hardware interfaces you have selected as the default settings. Use Debug Configurations if you want to edit or change your hardware interface and its settings. If the P&E Cyclone was selected, the USB port will be the default setting. Use Debug Configurations to switch to Ethernet or Serial port connections.

16.1.2 Launch Configurations Settings

To set the launch configurations for the debugger:

1. Right-click on your project and navigate to -> Debug As -> Debug Configurations. The Debug Configuration Window will appear.
2. In the left column, select the project type you would like to set the launch configurations.
3. In the right column, click the Main tab and use the Connection Selection drop-down box to select a connection.
4. Click the Edit button beside the selected connection and the Launch Configuration Window will appear.
5. Set your configurations, click the OK button, and click the Debug button to start the debugger.



Debug Configurations

Create, manage, and run configurations

Debug or run an application to a target.

type filter text

- CodeWarrior
  - S12ZVML128_FLASH_PnE U-MultiLink
- Launch Group

Filter matched 3 of 3 items

Filter by Project:

- 5676R
- CF51JF128
- K21DN512

Name: S12ZVML128_FLASH_PnE U-MultiLink

Debug session type

Choose a predefined debug session type or custom type for maximum flexibility

● Download ○ Connect

○ Attach ○ Custom

C/C++ application

Project: S12ZVML128

☑ Application: FLASH/S12ZVML128.elf

Build (if required) before launching

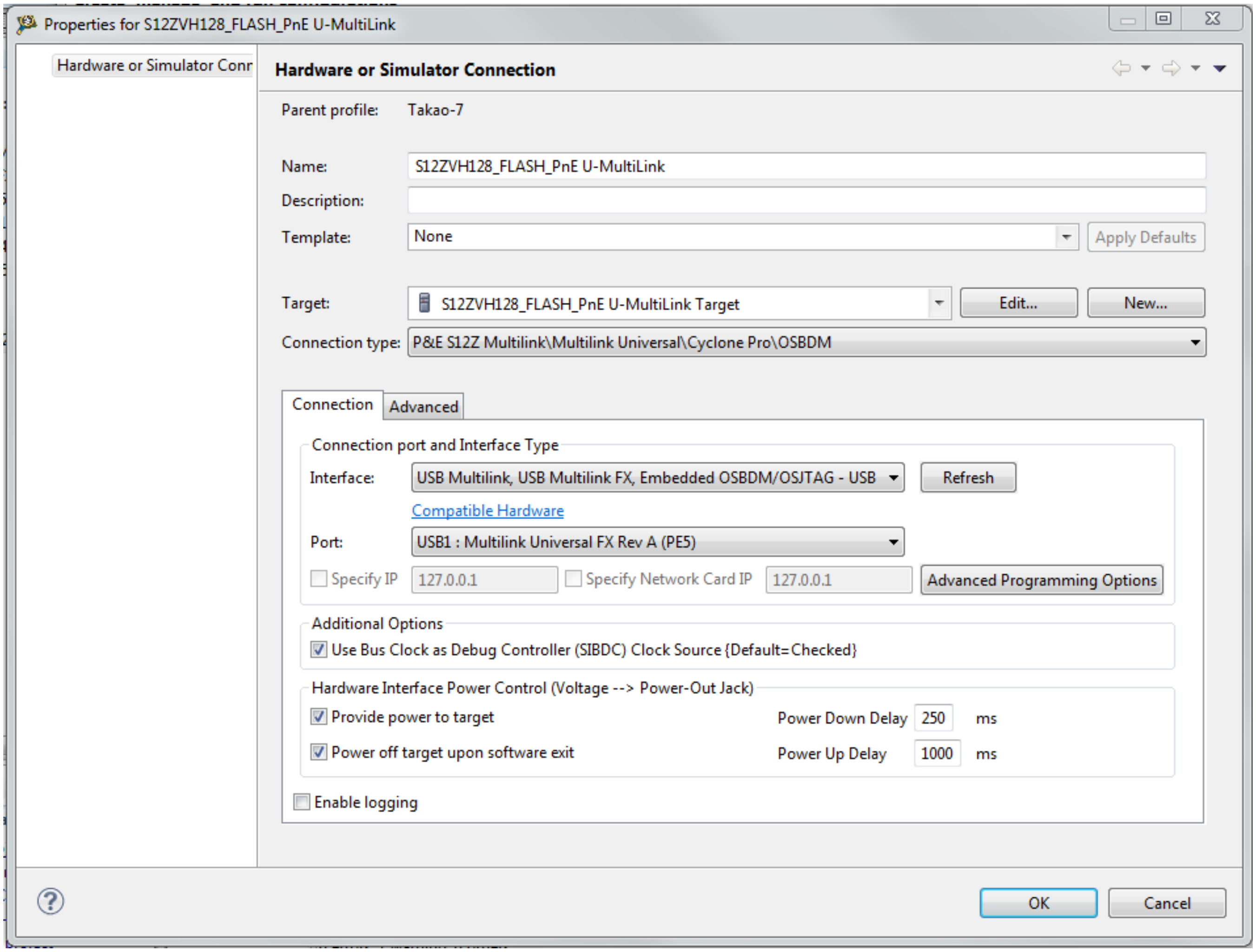
Target settings

Connection: S12ZVML128_FLASH_PnE U-MultiLink

☑ Execute reset sequence

☑ Execute initialization script(s)

Figure 16-2. Debug Configuration Dialog Box



Properties for S12ZVH128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVH128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVH128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

□ Enable logging

Figure 16-3. Launch Configuration Dialog Box

16.1.3 Connection Options

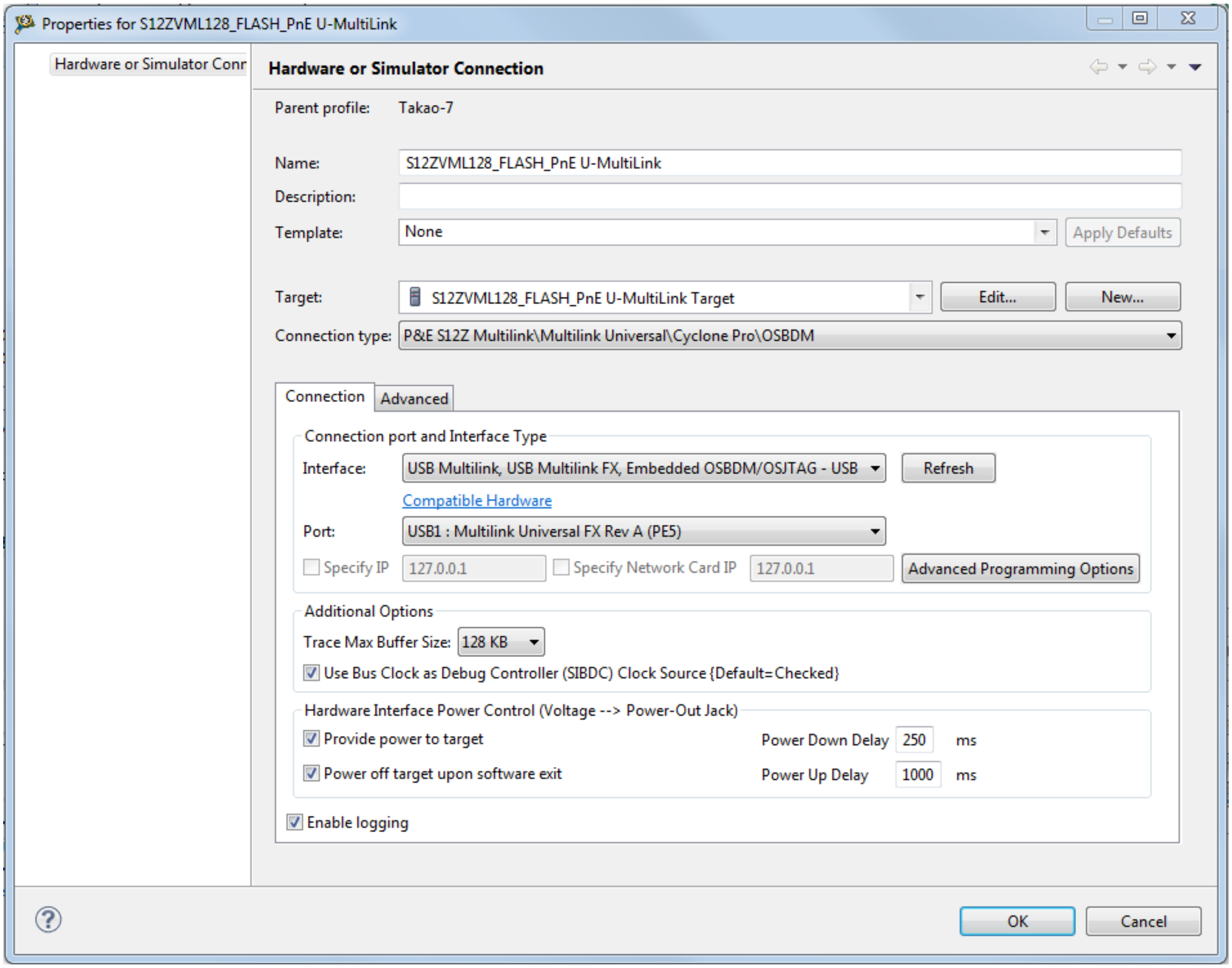
This topic describes all P&E Multilink/Cyclone/Tracelink/OSBDM connection options, which are common to all P&E USB Multilink Universal [FX]/USB Multilink, P&E Cyclone Serial, P&E Cyclone USB , P&E Cyclone Ethernet, P&E TraceLink USB, and P&E TraceLink Ethernet connections.

The options include:

- Changing P&E Connection Settings
- Connection Assistant

16.1.3.1 Changing P&E Connection Settings

All connection settings for P&E hardware interfaces are configured in the Launch Configurations dialog box.



Properties for S12ZVML128_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVML128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVML128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

☑ Enable logging

Figure 16-4. P&E S12Z Launch Configuration Dialog Box

The followin table describes the options for this view.

Table 16-1. Connection Parameter Options

Option	Description
Interface	Use this option to select the interface type. Select a supported interface from the list box. The options are: • USB BDM Multilink (HCS08/HCS12/CFV1)- USB Port • USB Multilink Universal [FX] - USB Port

Table continues on the next page...

Table 16-1. Connection Parameter Options (continued)

Option	Description
	NOTE: The USB Multilink Universal can conveniently support all Freescale architectures found in the current CodeWarrior 10 version • Cyclone- Serial Port • Cyclone- USB Port • Cyclone- Ethernet Port • Tracelink - USB Port • Tracelink - Ethernet Port • OSBDM NOTE: Click on the "Compatible Hardware" link to help you determine which P&E hardware is most suitable for your project.
Refresh	Click this button to have the workstation scan for a valid interface and port. Valid interfaces and ports appear in the Interface and Port list boxes.
Port	This option selects the port over which debug communications is conducted. Select an available port from the list box. NOTE: If you are having issues trying to get a port to display, click on the [FAQ #29] link for help.
Specify IP (Cyclone Ethernet only)	Use this option to specify the IP address of a Cyclone outside of the local network. Click on the checkbox to enable the textbox. This will also disable the port dropdown box. Currently supports IPv4 only.
Specify Network Card IP (Cyclone Ethernet only)	Use this option to specify the local network card IP address if there are multiple cards on your computer. Click on the checkbox to enable the textbox. Currently supports IPv4 only.
Provide power to target (Cyclone, Tracelink, and USB Multilink Universal FX only)	Check this option to have the Cyclone , Tracelink, or USB Multilink Universal FX (circuitry) supply power to the hardware target.. Uncheck this option to not provide power. **NOTE:** For USB Multilink Universal FX, use the jumper settings located at JP10 to provide either 3.3V or 5V.
Power off target upon software exit (Cyclone, Tracelink, and USB Multilink Universal FX only)	Check this option to turn off the power when the program terminates. Uncheck this option to leave the hardware target powered continuously.
Regulator Output Voltage (Cyclone, Tracelink, and USB Multilink Universal FX only)	This option adjusts the output voltage that powers the hardware target. Select a voltage value from this option's list box.
Power down delay (Cyclone, Tracelink, and USB Multilink Universal FX only)	This option specifies amount of time for which the target will be turned off during a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.
Power up delay (Cyclone, Tracelink, and USB Multilink Universal FX only)	This option specifies amount of time for which the target will remain powered prior to a RESET power cycling sequence. Enter the delay interval (in milliseconds) in this option's text box.

NOTE

Improper voltage setting can damage the board.

To change P&E Connections settings, perform these steps.

1. In the CodeWarrior Projects view, select the project for which you want to change the P&E Connections settings.

   NOTE

   It is assumed that you have created a project and built it.

2. Select Run > Debug Configurations from the main menu bar of the IDE.

   The Debug Configurations dialog box appears.

3. Expand the CodeWarrior tree control in the left pane and select the launch configuration you want to debug.
4. Click the Main tab.

   The Main page appears in the area beneath the tabs.

5. Select a system within Connection which you would like to use to debug. You could create a new system by clicking the New button. For more details about creating a new remote system, refer to the topic Target Management via Remote System Explorer in the CodeWarrior Common Features Guide. Once a remote system is selected, click the Edit button. The Launch Configuration Panel will appear.
6. Ensure that the Target is the correct microcontroller you want to debug. Use the drop-down box or the Edit button to change this option.
7. In the Connection Type drop-down box, select P&E S12Z Multilink/Multilink Universal/Cyclone/OSBDM. The P&E connections settings will appear below.
8. Click Refresh to scan valid interface and port.

   Valid interfaces and ports appear in the Interface and Port drop-down lists in the Connection Port and Interface Type group.

9. Select a supported interface from the Interface drop-down list.
10. Select a supported port from the Port drop-down list.

    NOTE

    The port displayed may vary depending on the interface.
    For example, if you select interface as Cyclone- Serial Port,
    the available port option is COM1 : Serial Port 1.

11. Specify settings in the Cyclone Power Control (Voltage --> Power -Out Jack) group.

    NOTE

    This group will be enabled enabled for the Cyclone,
    Tracelink and USB Multilink Universal FX interfaces only.
    For USB Multilink Universal FX interface, use the jumper
    settings located at JP10 to provide either 3.3V or 5V.

- Check the Provide power to target checkbox to have the hardware interface (circuitry) provide power to the target else clear the checkbox if you do not want to provide power to the target.
- Check the Power off target upon software exit checkbox to turn off the power when the program terminate else clear the checkbox to leave the hardware target powered continuously.
- Select a voltage value from the Regulator Output Voltage drop-down list. This adjusts the output voltage that powers the hardware target.

NOTE

An improper voltage setting can damage the board.

- Enter the delay interval (in milliseconds) in the Power Down Delay text box. This option specifies the time interval to wait before shutting off the power to the hardware target. The hardware interface powers down the device once the debug session is over, or while executing a power cycling sequence after beginning a new debug session.
- Enter the delay interval (in milliseconds) in the Power Up Delay text box. This option specifies the time interval to wait before turning on the power to the hardware target. If the power to target feature is enabled, the hardware interface will power up the device while executing a power cycling sequence at the beginning of every debug session.
- Click OK to save changes to the P&E Connections settings. The Launch Configuration Panel dialog box will close.
- Click Close button to close the Debug Configuration dialog box.

16.1.3.1.1 P&E Hardware Interface Connection-Specific Options

This topic describes the connection-specific options. The connections include:

- P&E USB Multilink Universal [FX]/USB Multilink
- P&E Cyclone Serial
- P&E Cyclone USB
- P&E Cyclone Ethernet
- P&E TraceLink USB
- P&E TraceLink Ethernet
- Open Source BDM

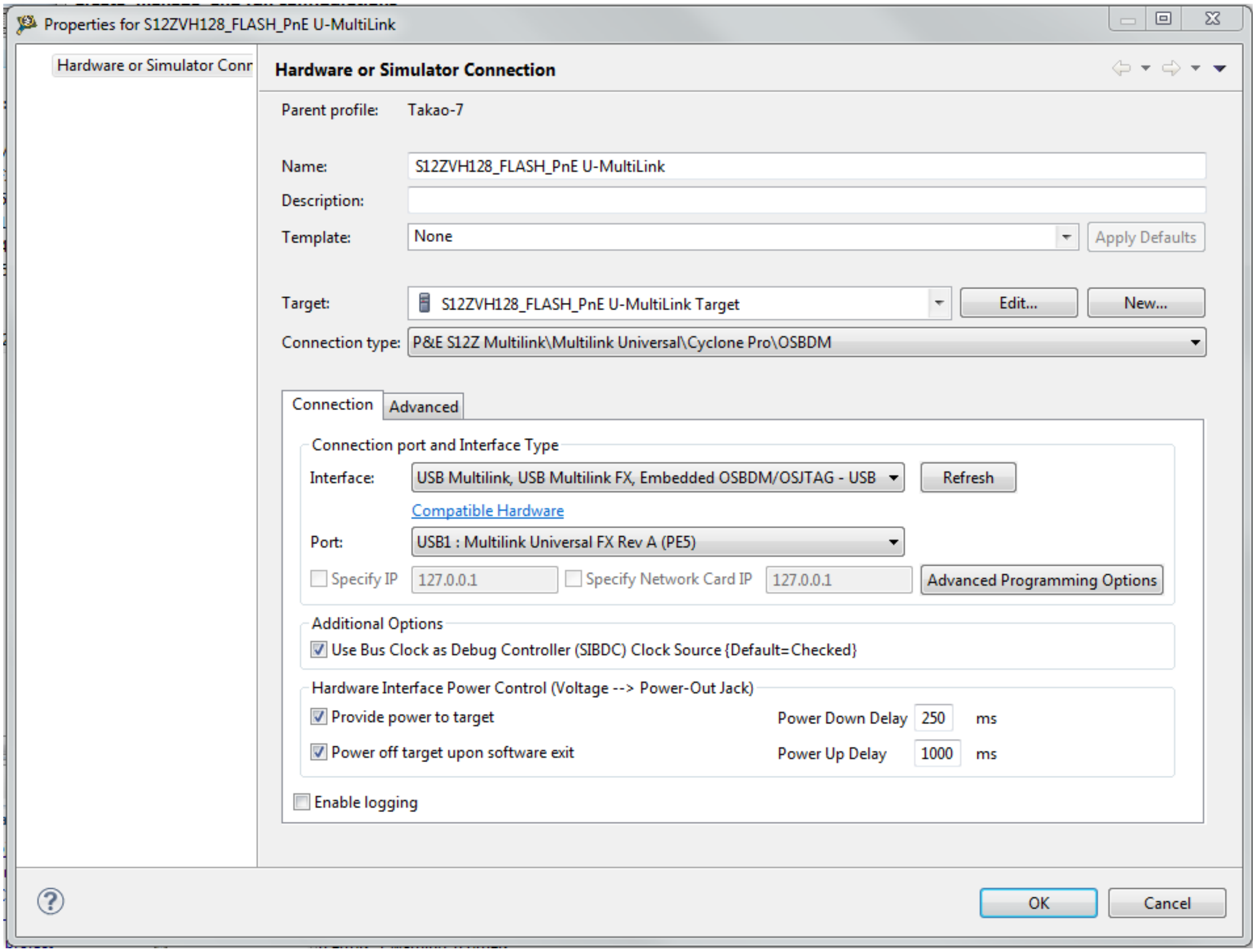
16.1.3.1.1.1 P&E USB Multilink Universal [FX]/USB Multilink

The P&E S12Z Connection setting permits a connection to USB Multilink devices, which include the P&E BDM Multilink, USB Multilink Universal, and the USB Multilink Universal FX. The P&E USB Multilink Universal [FX]/USB Multilink mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. Like the USB Multilink Universal, the USB Multilink Universal FX can conveniently debug all Freescale architectures found in the current CodeWarrior 10 version, however, the FX version is up to 8 times faster than the USB Multilink Universal and it can also provide power to the target.

16.1.3.1.1.1.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings.



Properties for S12ZVH128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVH128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVH128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

□ Enable logging

Figure 16-5. P&E's Launch Configuration Dialog Box

To use P&E's USB Multilink Universal [FX]/USB Multilink, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

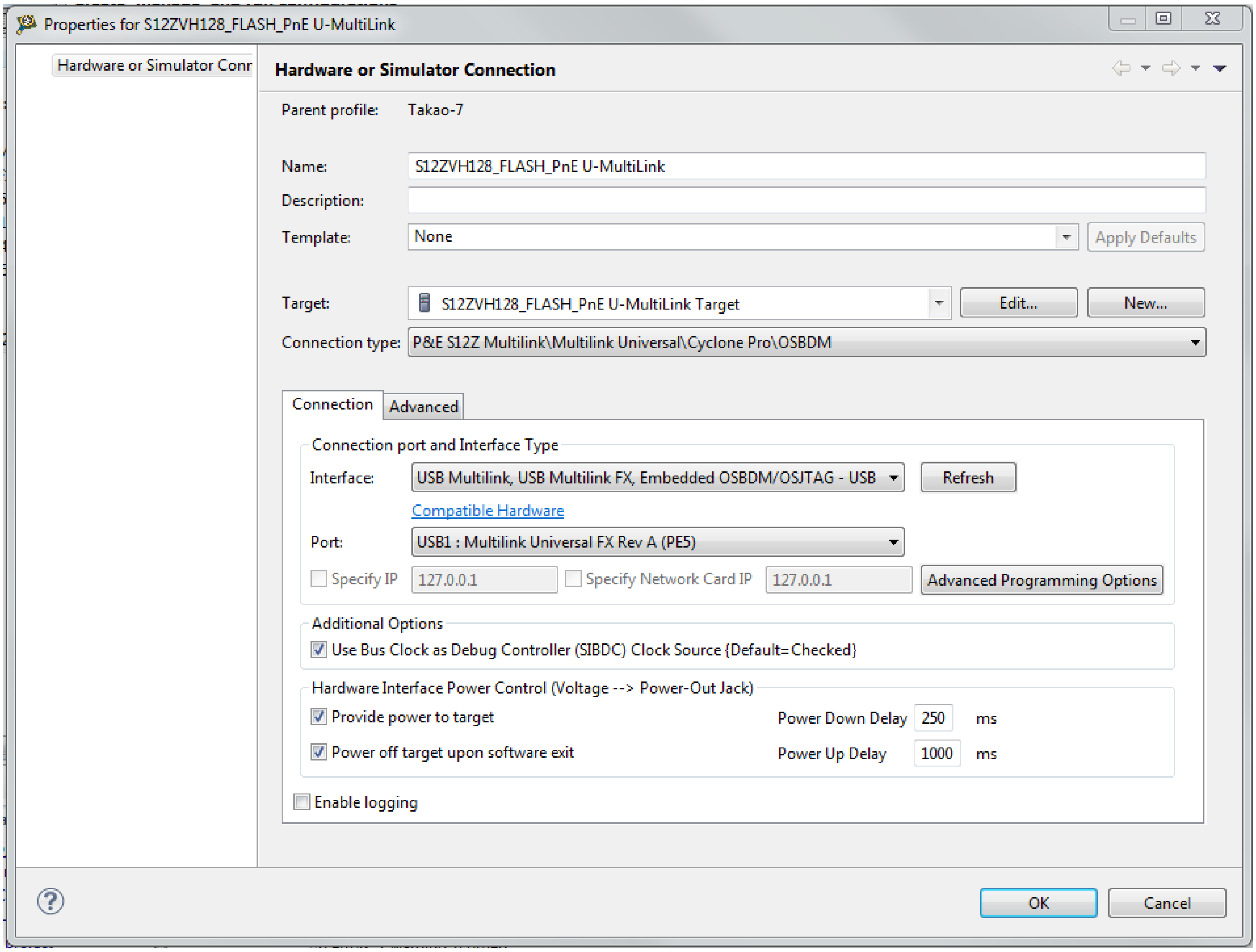
16.1.3.1.1.2 P&E Cyclone Serial

The P&E Cyclone Serial Connection setting permits a connection to Cyclone Serial devices. P&E Cyclone Serial mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

16.1.3.1.1.2.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for S12ZVH128_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVH128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVH128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

□ Enable logging

Figure 16-6. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone Serial, first connect your hardware interface to your computer, and then set the interface to Cyclone - Serial Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

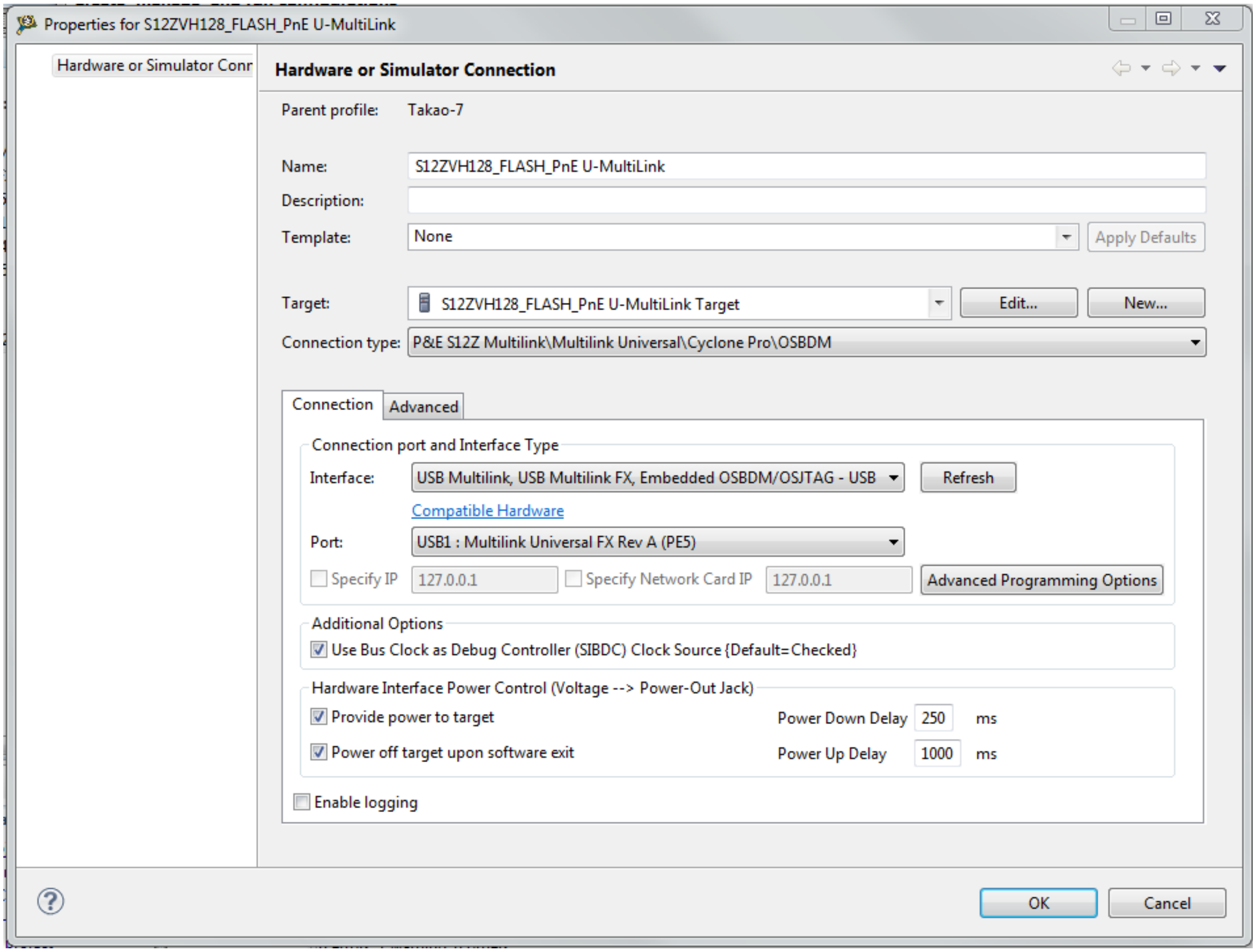
16.1.3.1.1.3 P&E Cyclone USB

The P&E Cyclone USB Connection setting permits a connection to Cyclone USB devices. P&E Cyclone USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

16.1.3.1.1.3.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/CyclonemPro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for S12ZVH128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVH128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVH128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

□ Enable logging

Figure 16-7. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone – USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

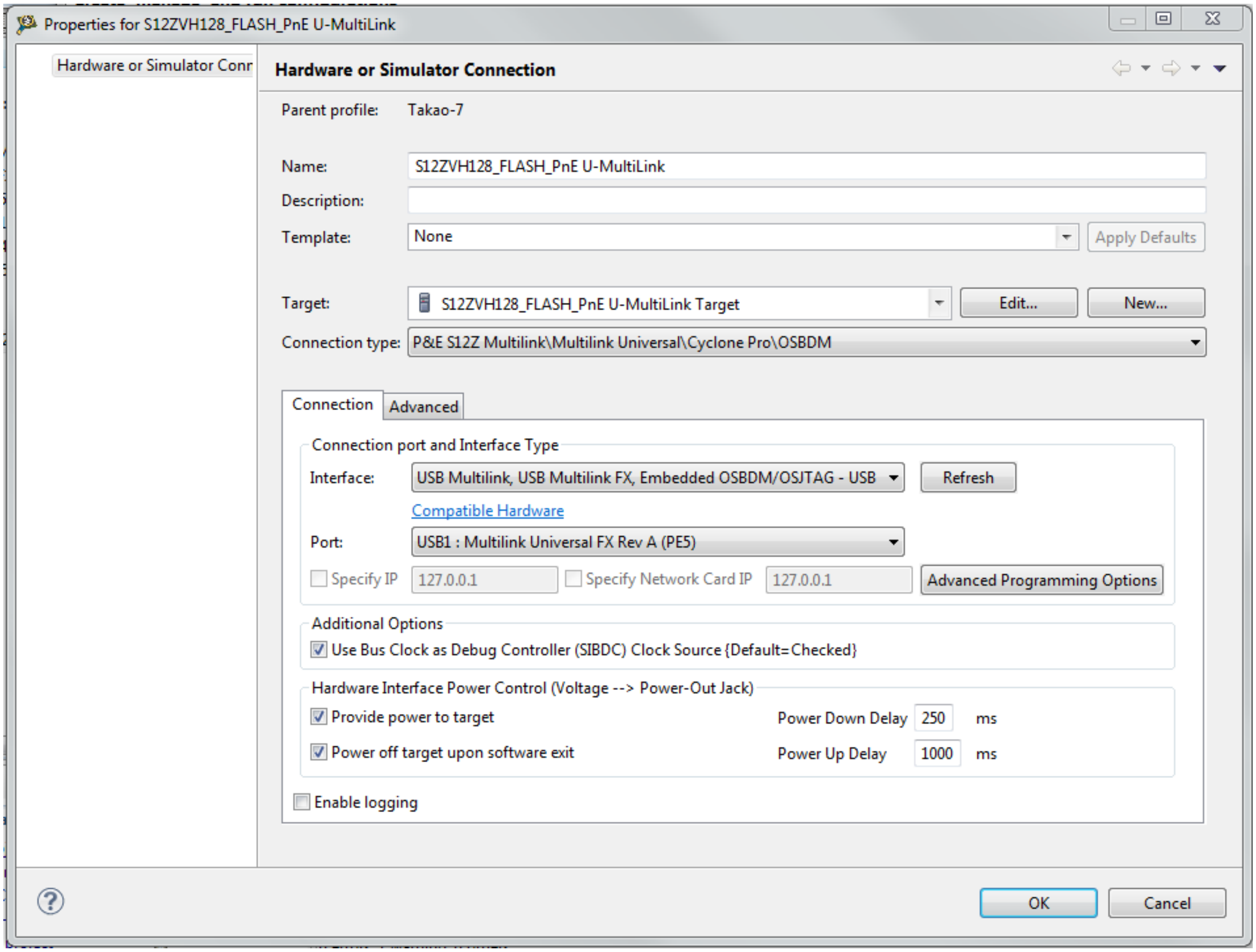
16.1.3.1.1.4 P&E Cyclone Ethernet

The P&E Cyclone Ethernet Connection setting permits a connection to Cyclone Ethernet devices. P&E Cyclone Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

16.1.3.1.1.4.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/Cyclone Pro/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for S12ZVH128_FLASH_PnE U-MultiLink

Hardware or Simulator Conn

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVH128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVH128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Compatible Hardware

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source (Default=Checked)

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target Power Down Delay 250 ms

☑ Power off target upon software exit Power Up Delay 1000 ms

□ Enable logging

Figure 16-8. P&E's Launch Configuration Dialog Box

To use P&E’s Cyclone Ethernet, first connect your hardware interface to your computer, and then set the interface to Cyclone – Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the

Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

16.1.3.1.1.5 P&E TraceLink USB

The P&E TraceLink USB Connection setting permits a connection to TraceLink USB devices. P&E TraceLink USB mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. P&E's TraceLink can conveniently support all Freescale architectures found in the current CodeWarrior 10 version; however, it can only capture the external trace signals for the ColdFire V2/3/4 and Kinetis ARM architectures.

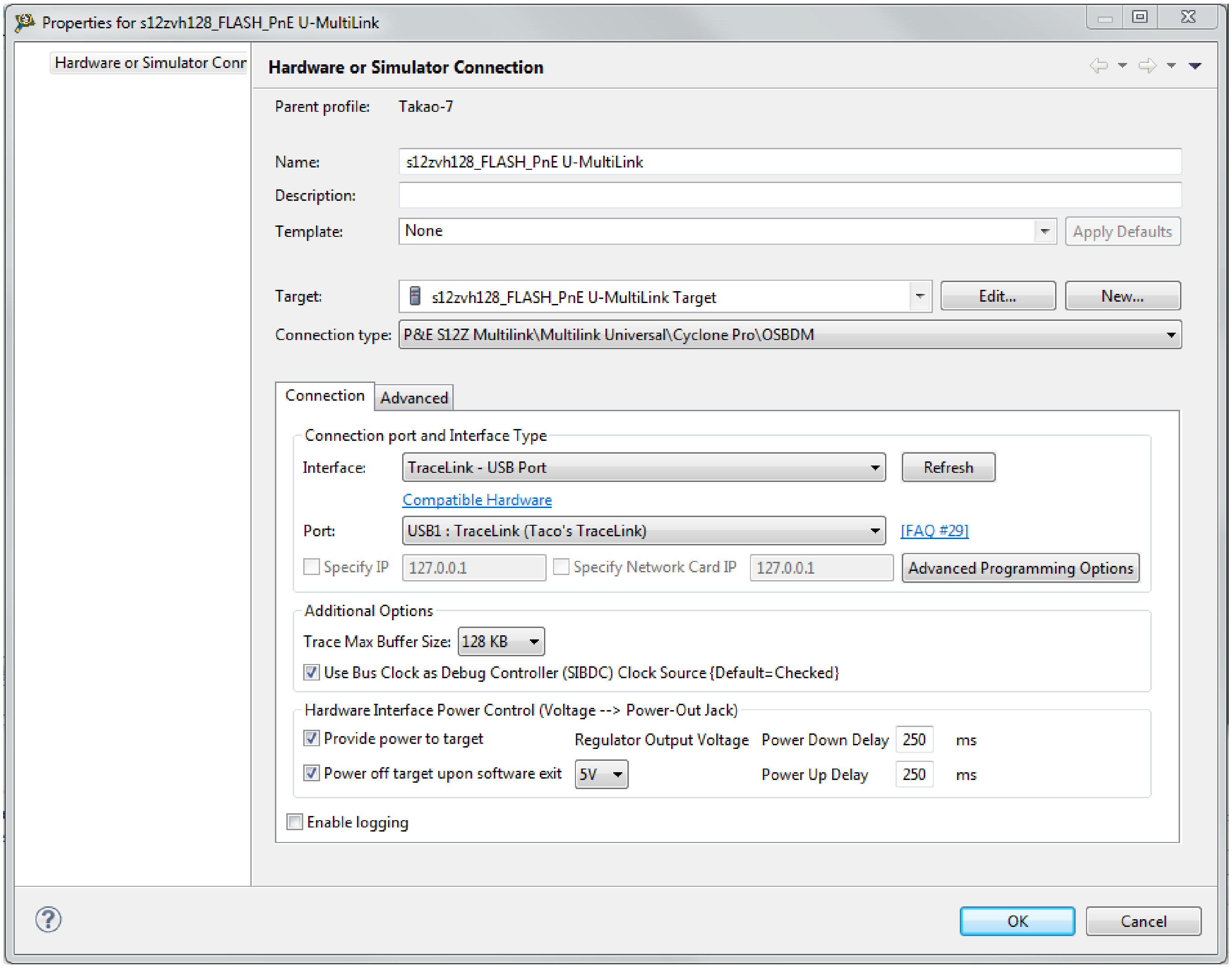
NOTE

S12Z devices can be debugged by the Tracelink, however Trace and Profile is currently not supported.

16.1.3.1.1.5.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear.
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for s12zvh128_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: s12zvh128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: s12zvh128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: TraceLink - USB Port

Compatible Hardware

Port: USB1 : TraceLink (Taco's TraceLink)

[FAQ #29]

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

Trace Max Buffer Size: 128 KB

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target

Regulator Output Voltage 5V

Power Down Delay 250 ms

☑ Power off target upon software exit

Power Up Delay 250 ms

□ Enable logging

Figure 16-9. P&E's Launch Configuration Dialog Box

To use P&E's Cyclone USB, first connect your hardware interface to your computer, and then set the interface to Cyclone - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

16.1.3.1.1.6 P&E TraceLink Ethernet

The P&E TraceLink Ethernet Connection setting permits a connection to TraceLink Ethernet devices. P&E TraceLink Ethernet mode lets you debug code, as the firmware is fully resident in the Flash of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources. P&E’s TraceLink can conveniently support all Freescale architectures found in the current CodeWarrior 10 version; however, it can only capture the external trace signals for the ColdFire V2/3/4 and Kinetis ARM architectures

NOTE

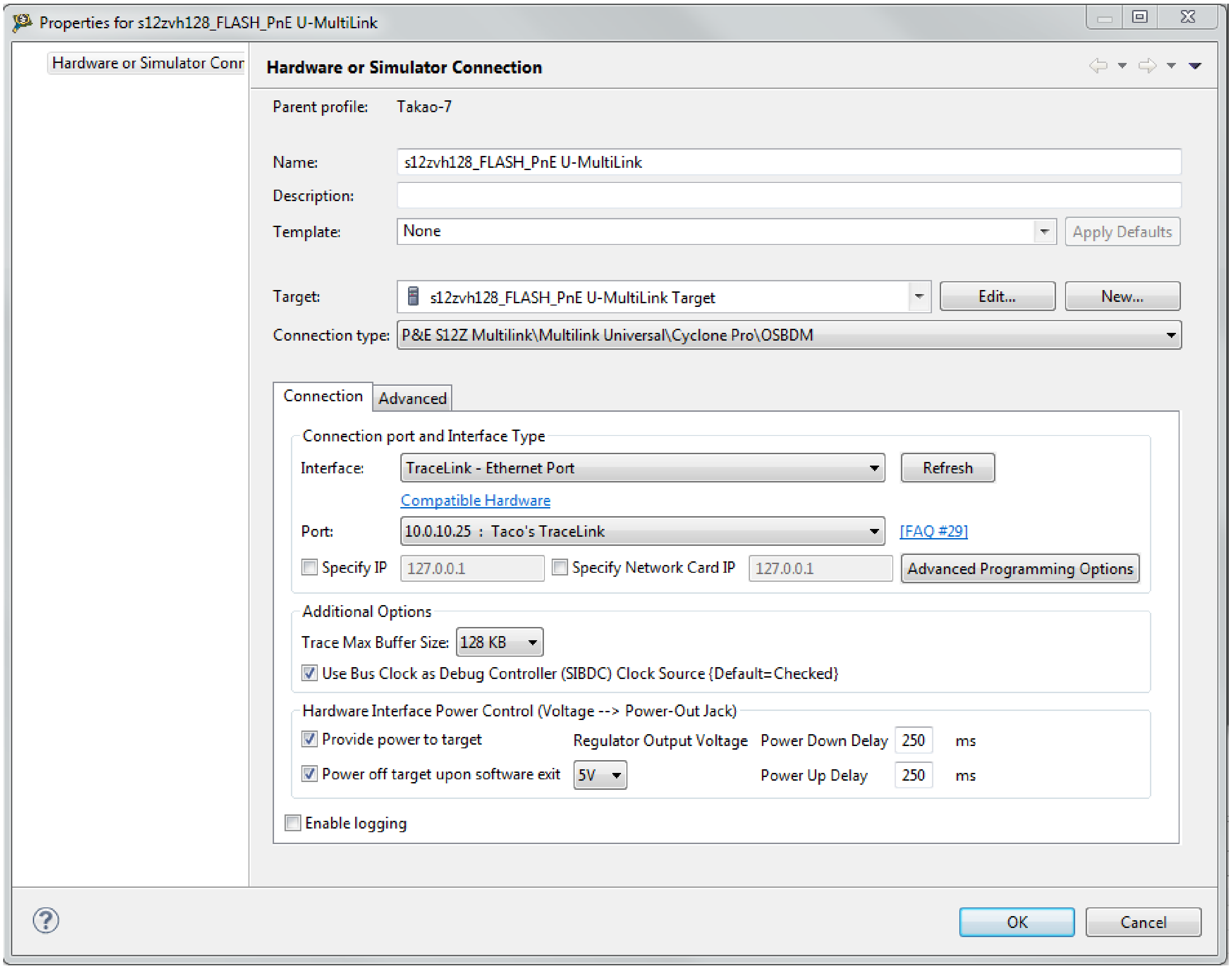
S12Z devices can be debugged by the Tracelink, however Trace and Profile iscurrently not supported.

.

16.1.3.1.1.6.1 Debug configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration dialog box will appear.
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/Cyclone/ OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for s12zvh128_FLASH_PnE U-MultiLink

Hardware or Simulator Conr

Hardware or Simulator Connection

Parent profile: Takao-7

Name: s12zvh128_FLASH_PnE U-MultiLink

Description:

Template: None   Apply Defaults

Target: s12zvh128_FLASH_PnE U-MultiLink Target   Edit...   New...

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection | Advanced

Connection port and Interface Type

Interface: TraceLink - Ethernet Port   Refresh

Compatible Hardware

Port: 10.0.10.25 : Taco's TraceLink   [FAQ #29]

☐ Specify IP 127.0.0.1   ☐ Specify Network Card IP 127.0.0.1   Advanced Programming Options

Additional Options

Trace Max Buffer Size: 128 KB

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target   Regulator Output Voltage   Power Down Delay 250 ms

☑ Power off target upon software exit   5V   Power Up Delay 250 ms

☐ Enable logging

OK   Cancel

Figure 16-10. P&E's Launch Configuration Dialog Box

To use P&E's TraceLink Ethernet, first connect your hardware interface to your computer, and then set the interface to TraceLink – Ethernet Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. You can also specify IP and Network Card IP by clicking on the checkboxes. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

16.1.3.1.1.7 Open Source BDM

Freescale supplies certain development boards with an integrated debug circuit based on Open Source BDM. This allows the development board to be debugged from the PC via the USB bus without requiring external debug hardware, such as the Cyclone or USB Multilink Universal [FX]/USB Multilink. The development board also derives its power from the USB Bus.

The Open Source BDM circuit design (OSBDM-JM60) is an open source, community driven design. It has been published on Freescale's website, and full documentation can be found in the Community Forums. The latest documentation and firmware can be downloaded from www.pemicro.com/osbdm.

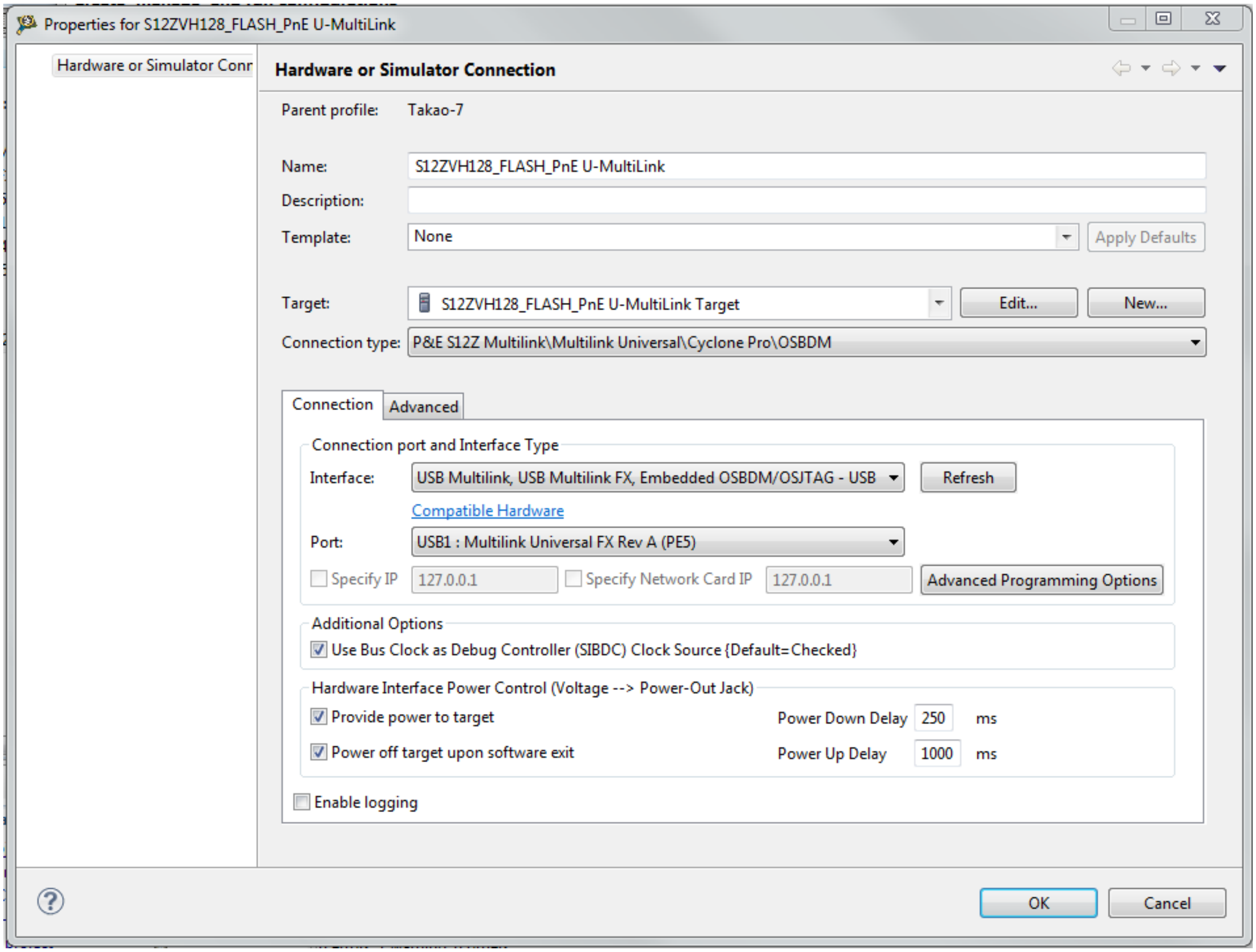
Integration with CodeWarrior is handled via the "Open Source BDM" connection. P&E has integrated the Open Source BDM support into the same connection that supports both the USB Multilink and the Cyclone. All of the dialogs that affect operation of these hardware interfaces function in the same manner when using OSBDM (albeit at a lower data rate).

The Open Source BDM Connection setting permits a connection to Open Source BDM devices. Open Source BDM mode lets you debug code, as the firmware is fully resident in the Flash or RAM of the microprocessor. The operation of all modules fully reflects the actual operation of the onboard resources.

16.1.3.1.1.7.1 Debug Configurations

To change P&E's hardware interface connection settings within Debug Configurations:

1. In the left column select the project for which you would like to change the settings. Within the Main tab, use the Connection selection drop-down box to select a connection.
2. Then click on the Edit button beside your selected connection. The Launch Configuration Dialog will appear. \
3. Set the connection type to P&E S12Z Multilink/Multilink Universal/Cyclone/OSBDM so that the Connection tab will populate P&E's hardware interface connection settings, as shown in the following figure.



Properties for S12ZVH128_FLASH_PnE U-MultiLink

Hardware or Simulator Connection

Parent profile: Takao-7

Name: S12ZVH128_FLASH_PnE U-MultiLink

Description:

Template: None

Target: S12ZVH128_FLASH_PnE U-MultiLink Target

Connection type: P&E S12Z Multilink\Multilink Universal\Cyclone Pro\OSBDM

Connection port and Interface Type

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB

Port: USB1 : Multilink Universal FX Rev A (PE5)

□ Specify IP 127.0.0.1 □ Specify Network Card IP 127.0.0.1

Additional Options

☑ Use Bus Clock as Debug Controller (SIBDC) Clock Source {Default=Checked}

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☑ Provide power to target — Power Down Delay 250 ms

☑ Power off target upon software exit — Power Up Delay 1000 ms

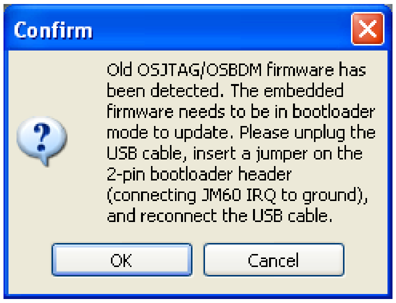
□ Enable logging

Figure 16-11. P&E's Launch Configuration Dialog Box

To use Open Source BDM, first connect your hardware interface to your computer, and then set the interface to USB Multilink, USB Multilink FX, Embedded OSBDM/ OSJTAG - USB Port. The Port selection should automatically populate your hardware interface. If not, use the Refresh button and the Port selection drop-down box. If you are having issues getting your device to populate, use the link for FAQ #29 to find popular solutions.

16.1.3.1.1.7.2 OSBDM Firmware Update

All CodeWarrior IDE's version 10.1 and higher have an automatic firmware update mechanism for built-in OSBDM hardware interfaces. Whenever an OSBDM-integrated hardware interface is plugged into a USB port and CodeWarrior attempts to contact the device, it will automatically check to see if the device has the latest OSBDM firmware version. If the firmware on the device is older than the one found within the CodeWarrior package, then a dialog box will indicate that a firmware update is necessary.

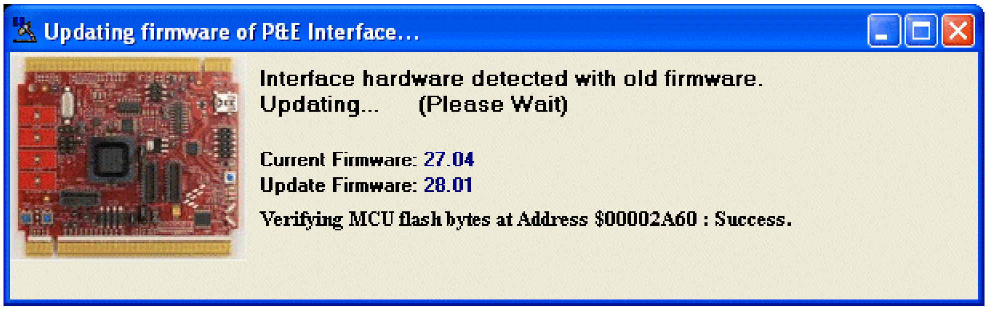


Confirm

Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.

Figure 16-12. Old OSBDM Firmware Detected

To update the firmware, the OSBDM device must enter Bootloader mode. To do so the USB cable must be disconnected from the device and the OSBDM-JM60 IRQ pin must be connected to ground usually done by using a 2-pin female jumper. Use the OSBDM device schematics to find the IRQ pin. Once the IRQ pin is grounded, connect the USB cable to the OSBDM device and click on the OK button. If done correctly, the automatic firmware update will occur.



Updating firmware of P&E Interface...

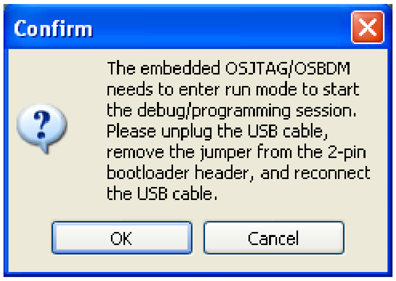
Interface hardware detected with old firmware.
Updating... (Please Wait)

Current Firmware: 27.04
Update Firmware: 28.01

Verifying MCU flash bytes at Address $00002A60 : Success.

Figure 16-13. OSBDM Firmware Updating

When the firmware is done updating, a dialog box will indicate that the OSBDM device must exit Bootloader mode and enter into Run mode.



Confirm

The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.

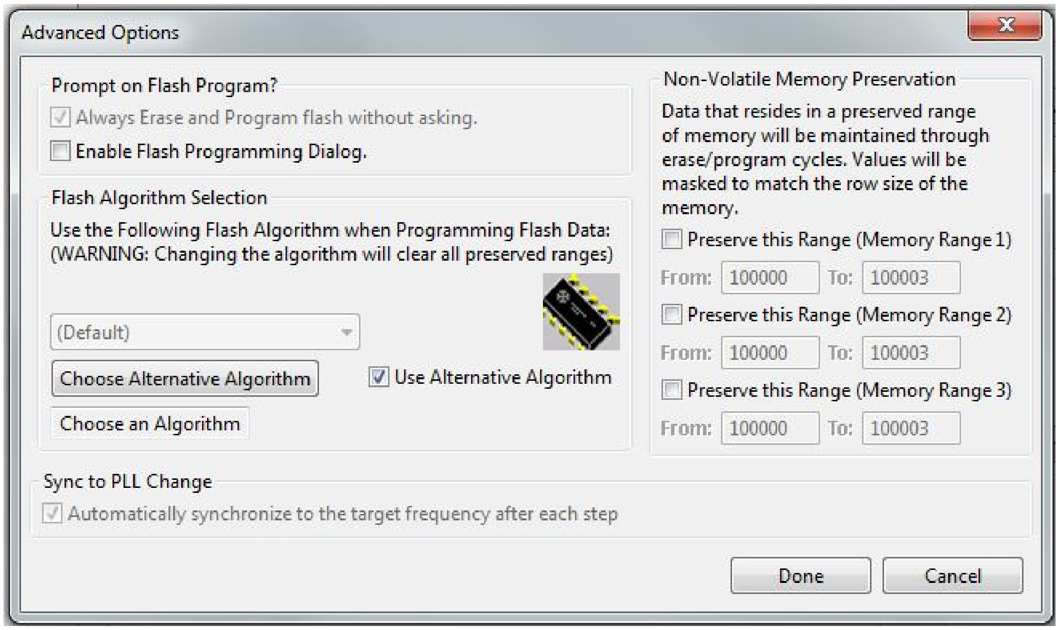
Figure 16-14. Start OSBDM Run Mode

To enter Run Mode, the user must disconnect the USB cable from the OSBDM device and the 2-pin female jumper on the IRQ pin must be removed. Next, reconnect the USB cable and the device will be in Run Mode. Click on OK and CodeWarrior will move onto programming or running the code.

The CodeWarrior IDE layout will have the latest OSBDM firmware. If for any reason you experience difficulty performing OSBDM firmware update, visit www.pemicro.com/osbdm and use the Multilink/OSBDM Firmware Update Utility to force an update, or use the OSBDM Firmware Recovery Utility for a fail safe way to reprogram a working, corrupted, or blank OSBDM firmware via an external USB-ML-12 hardware interface.

16.1.3.1.2 Advanced Programming/Debug Options

The Advanced Programming/Debug Options menu option takes you to the Advanced Options dialog box, where you can configure the software settings for the flash programming procedure.



Advanced Options

Prompt on Flash Program?

☑ Always Erase and Program flash without asking.

☐ Enable Flash Programming Dialog.

Flash Algorithm Selection

Use the Following Flash Algorithm when Programming Flash Data:
(WARNING: Changing the algorithm will clear all preserved ranges)

(Default)

Choose Alternative Algorithm ☑ Use Alternative Algorithm

Choose an Algorithm

Non-Volatile Memory Preservation

Data that resides in a preserved range of memory will be maintained through erase/program cycles. Values will be masked to match the row size of the memory.

☐ Preserve this Range (Memory Range 1)

From: 100000 To: 100003

☐ Preserve this Range (Memory Range 2)

From: 100000 To: 100003

☐ Preserve this Range (Memory Range 3)

From: 100000 To: 100003

Sync to PLL Change

☑ Automatically synchronize to the target frequency after each step

Figure 16-15. Advanced Options Dialog Box

16.1.3.1.2.1 Enable Flash Programming Dialog

Setting the Enable Flash Programming dialog box lets you view the steps taken by the Flash Programmer.

16.1.3.1.2.2 Non-Volatile Memory Preservation

You have the option of preserving up to three independent ranges of non-volatile memory (on devices with EEPROM, the EEPROM array may optionally be preserved as well). Ranges that are designated as "preserved" are read before an erase and reprogrammed immediately afterwards, thereby preserving the data in these ranges. Any attempt to program data into a preserved range is ignored. When entering an address into the preserved range field (hexadecimal input is expected), the values are masked according to the row size of the device. This ensures that the reprogramming of preserved data does not cause any conditions that disturb programming.

16.1.3.1.2.3 Alternative Algorithm Functionality

Once you create a project for a specific S12Z microprocessor, the debugger specifies a default algorithm to use during all flash programming operations. The debugger uses this algorithm for nearly all programming requirements. The default algorithm can be found in the <CW_Install>/MCU/bin/plugins/support/S12Z/gdi/P&E directory

However, you can override the default algorithm via the Alternative Algorithm function, located in the Advanced Programming/Debug Options menu. This feature can be used to select a custom programming algorithm or select another one of P&E's many programming algorithms for use with a specific project.

CAUTION

> Selecting the wrong programming algorithm may damage your device, lead to under/over programming situations, or simply not program portions of the project file. Therefore it is recommended to use the default algorithm unless there is a compelling reason to do otherwise.

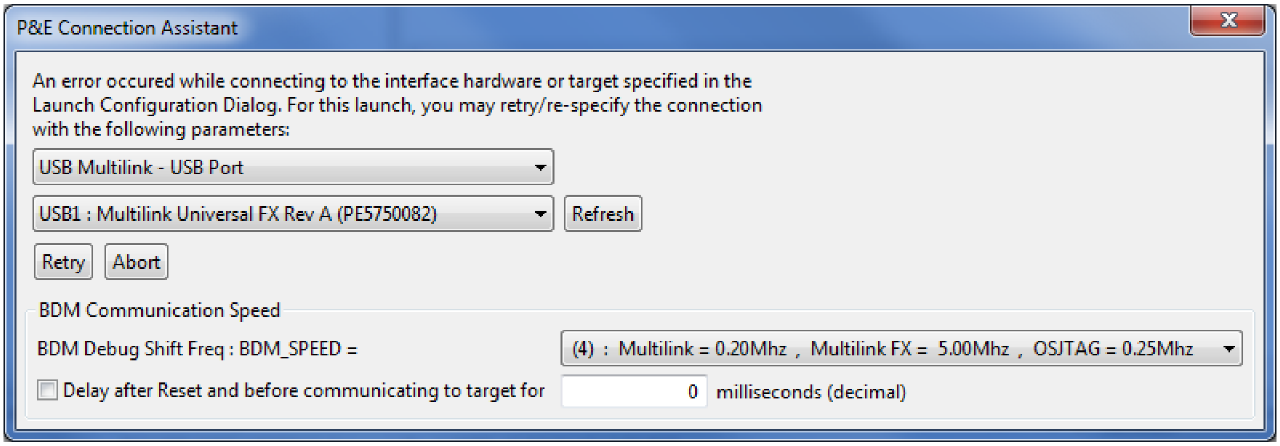
Use these steps to override the default algorithm:

1. Check the Use Alternative Algorithm checkbox.
2. Click the Choose Alternative Algorithm button, which lets you browse for an alternative algorithm.
3. Once you select the alternative algorithm, the name of the algorithm along with its full path appears in the text field below the Choose Alternative Algorithm button.
4. At this point, the current project will perform all future flash programming operations using the alternative algorithm. You may revert to the default algorithm at any time by clearing the Use Alternative Algorithm checkbox.

16.1.3.2 Connection Assistant

The P&E Connection Assistant is displayed when you attempt to debug and the program cannot connect to the interface hardware specified in the Launch Configuration dialog box. To select the P&E USB Multilink Universal [FX]/USB Multilink as your debugger connection:

1. Select USB Multilink - USB Port from the first drop-down menu and click **Refresh**.
2. Using the second drop-down menu, select the port on which the interface is connected.
3. Use the Cyclone Power Control panel to configure the power and delay settings (Cyclone Pro only).
4. Click the Retry button.


P&E Connection Assistant

An error occured while connecting to the interface hardware or target specified in the Launch Configuration Dialog. For this launch, you may retry/re-specify the connection with the following parameters:

USB Multilink - USB Port

USB1 : Multilink Universal FX Rev A (PE5750082)

BDM Communication Speed

BDM Debug Shift Freq : BDM_SPEED = (4) : Multilink = 0.20Mhz , Multilink FX = 5.00Mhz , OSJTAG = 0.25Mhz

☐ Delay after Reset and before communicating to target for 0 milliseconds (decimal)

Figure 16-16. S12Z Connection Assistant Interface Selected

16.1.4 Active Mode Menu Options

When the microprocessor is connected, the active mode menu shows the name of the microprocessor and gives you the access to P&E Microcomputer Systems' Compatible Hardware Interfaces web page and home page. If the OSBDM interface is being used for the debugging session, then the OSBDM Documentation option appears within the active mode menu, which takes you to P&E Microcomputer Systems' OSBDM website. When the microprocessor is not connected, the menu is not available.

Chapter 17
Common Connection Features

This chapter explains how to use the CodeWarrior hardware tools. Use these tools for board bring-up, test, and analysis.

The topics in this chapter are:

- Working with Flash Programmer
- Flash Programmer Tutorials
- Working with Hardware Diagnostics Window
- Manipulating Target Memory

17.1 Working with Flash Programmer

The CodeWarrior Flash Programmer can program the flash memory of the target board with code from any CodeWarrior IDE project or any individual files. The Flash Programmer (FP) feature is a target task that lets you run a series of actions on a flash: internal or present on a board (NOR, NAND, etc.). The supported operations are:

- **Program/Verify**, refer Add Program / Verify Actions.
- **Erase/Blank Check**, refer Add Erase/Blank Check Actions.
- **Checksum**, refer Add Checksum Actions.
- **Diagnostics**, refer Add Diagnostics Actions.
- **Dump Flash**, refer Add Dump Flash Actions.
- **Protect/Unprotect**, refer Add Protect/Unprotect Actions.
- **Secure/Unsecure**, refer Add Secure/Unsecure Actions

The CodeWarrior Flash Programmer lets you program the flash memory of any of the supported target boards, from within the IDE. You can do this using either the predefined tasks provided with the CodeWarrior installation, or create your own specialized tasks. Each of these options is described in the following topics:

- Use Pre-Defined Programming Task
- Create Flash Programmer Task
- Flash Programmer for MCF5441X
- Importing Pre-defined Task
- Creating Flash Programmer Target Task
- Configure the Flash Programmer Target Task
- Run Flash Programmer Target Task

17.1.1 Use Pre-Defined Programming Task

To use a pre-defined flash programming task, you first import its `*.xml` file into the **Target Tasks** view. CodeWarrior for Microcontrollers provides default flash configuration files for a wide variety of supported target boards.

The pre-defined task files are in the following directories.

Table 17-1. Pre-defined Task Files Locations

Connection	Location
ARM	`<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/ARM`
ColdFire	`<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/ColdFire`
DSC	`<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/DSC`
E200	`<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/E200`
RS08	`<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/RS08`
S12Z	`<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/S12Z`

17.1.2 Importing Pre-defined Task

After you have imported the task, it appears in the **Target Tasks** view, where you can execute it.

To import a pre-defined Flash Programmer task:

1. From the CodeWarrior menu bar, select **Window > Show View > Other**.

The **Show View** dialog box appears.

2. Expand the **Debug** tree control and select **Target Tasks**.

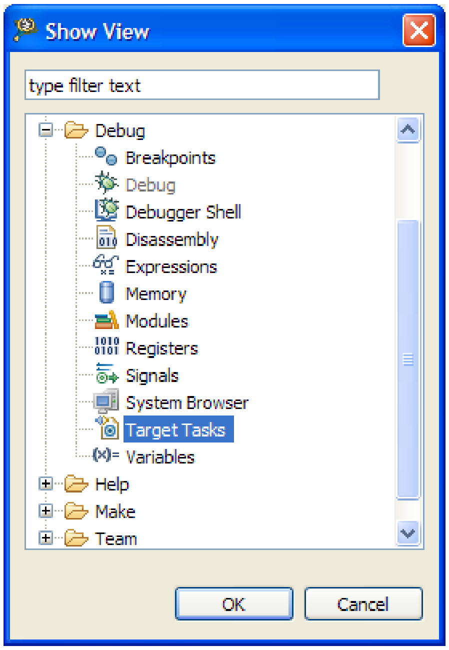


Figure 17-1. Show View Dialog Box

3. Click **OK**.

The **Target Tasks** view appears.

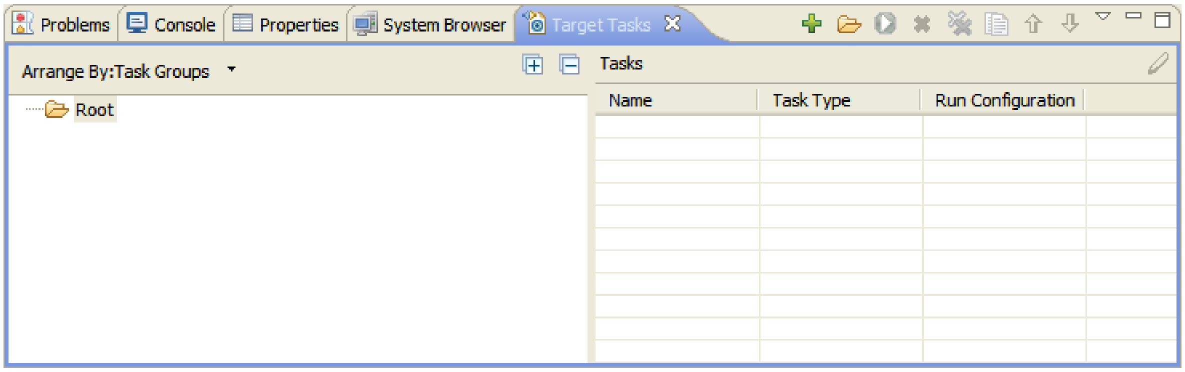


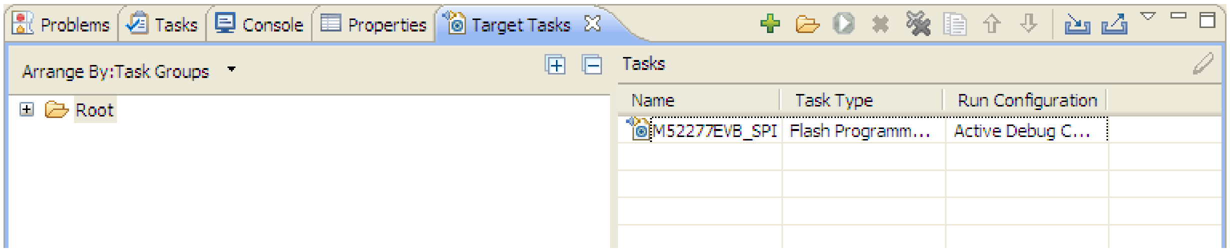
Figure 17-2. Target Tasks View

4. Right-click in the **Target Tasks** view and select **Import** . Alternatively, click the **Import** icon on the **Target Tasks** view toolbar.

The **Open** dialog box appears.

5. Navigate to the pre-defined tasks folder at `<CW MCU install>\MCU\bin\plugins\support\TargetTask\Flash_Programmer\` and select the desired `.xml` file for your hardware target. For example, select `MCF5213_INTFLASH.xml` from the `ColdFire` folder.
6. Click **Open**.

The selected task appears in the **Target Tasks** view.



Name	Task Type	Run Configuration
M52277EVB_SPI	Flash Programm...	Active Debug C...

Figure 17-3. Pre-defined Task in Target Tasks View

7. Right-click on the task's name and select **Execute**.

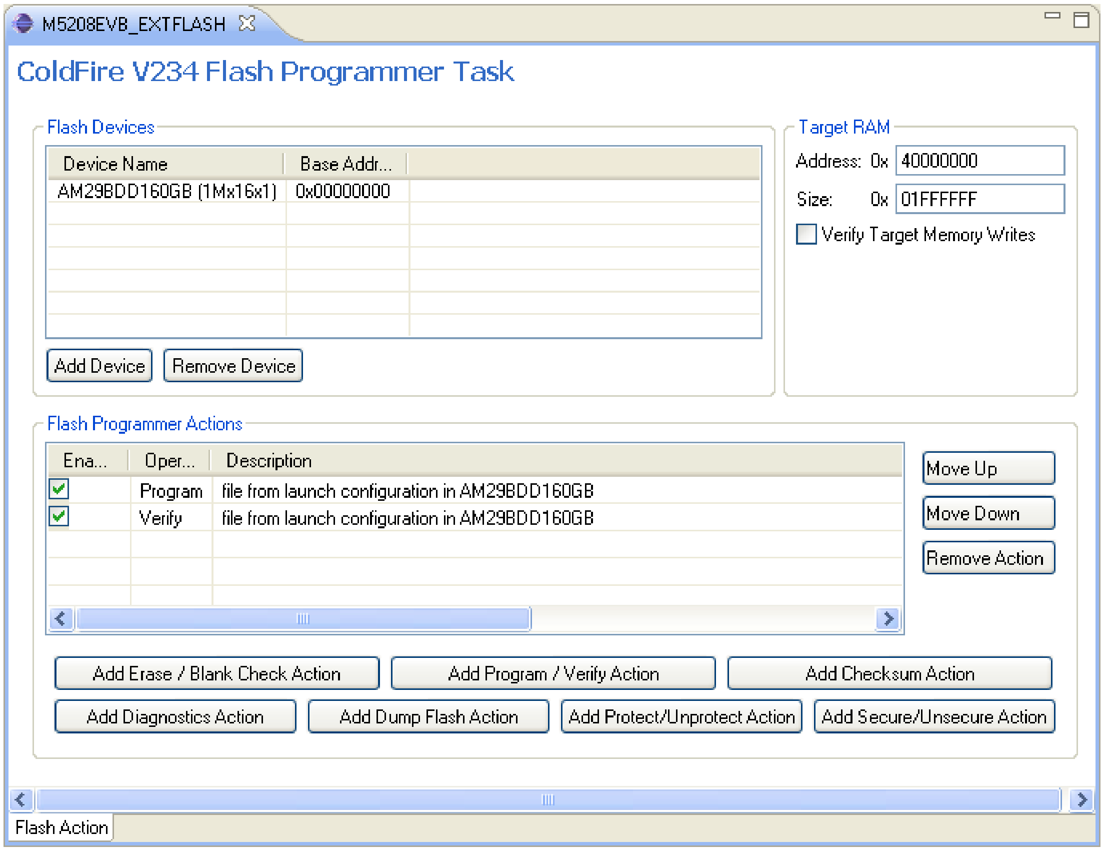
   The task's Flash Programmer actions execute in sequence. First, they erase the hardware target's flash memory.

   NOTE

   When a predefined flash programmer task is imported, its **Run Configuration** is set as `Active Debug Context`. If the task is imported and there is no active debug session, then the **Execute** icon will be disabled. Associate the selected target task to a different Run Configuration to enable the **Execute** icon.

8. Double-click on the task's name, to examine the task's stored Flash Programmer actions.

   The <target> **Flash ProgrammerTask** editor window appears, and displays the actions in the **Flash Programmer Actions** group.



M5208EVB_EXTFLASH

ColdFire V234 Flash Programmer Task

Flash Devices

Device Name	Base Addr...
AM29BDD160GB (1Mx16x1)	0x00000000

Target RAM

Address: 0x 40000000

Size: 0x 01FFFFFF

□ Verify Target Memory Writes

Flash Programmer Actions

Ena...	Oper...	Description
☑	Program	file from launch configuration in AM29BDD160GB
☑	Verify	file from launch configuration in AM29BDD160GB

Figure 17-4. <target> Flash Programmer Task Editor Window Displaying Stored Actions

If you are working with special hardware that require a different sequence of Flash Programmer actions, you can create your own target tasks.

17.1.3 Creating Flash Programmer Target Task

1. Select **Windows > Show Views > Others** from the IDE menu bar.

   The **Show View** dialog box appears.

2. Expand the **Debug** group and select **Target Tasks**.
3. Click **OK**.

   The **Target Tasks** view appears.

4. Click the **Create a new Target Task** icon in the **Target Tasks** view toolbar.

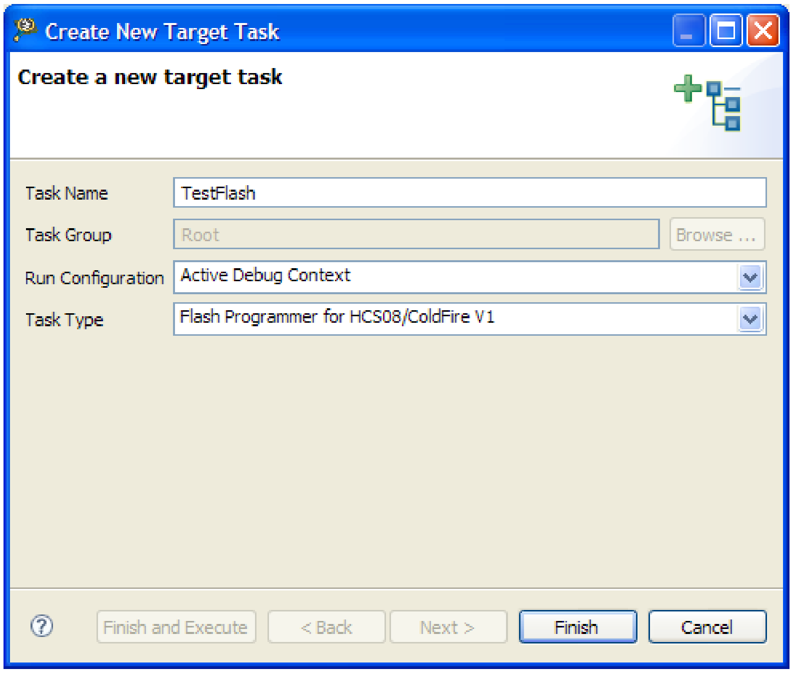
   The **Create New Target Task** wizard appears.

5. In the **Task Name** text box, enter the name of the target task.
6. From the **Run Configuration** drop-down list, select a configuration.

NOTE

Select **Active Debug Context** from the **Run Configuration** drop-down list, if you want to use Flash Programmer over an active debug session, else select any of the specified debug context from the list.

7. From the **Task Type** drop-down list, select the appropriate **Flash Programmer**.



Create New Target Task

Create a new target task

Task Name: TestFlash

Task Group: Root

Run Configuration: Active Debug Context

Task Type: Flash Programmer for HCS08/ColdFire V1

Figure 17-5. Create New Target Task Wizard

8. Click **Finish**.

   The <target> **Flash Programmer Task** editor window appears.

NOTE

The <target> **Flash Programmer Task** editor window has groups to define flash devices, Flash Programmer actions, and target RAM settings.

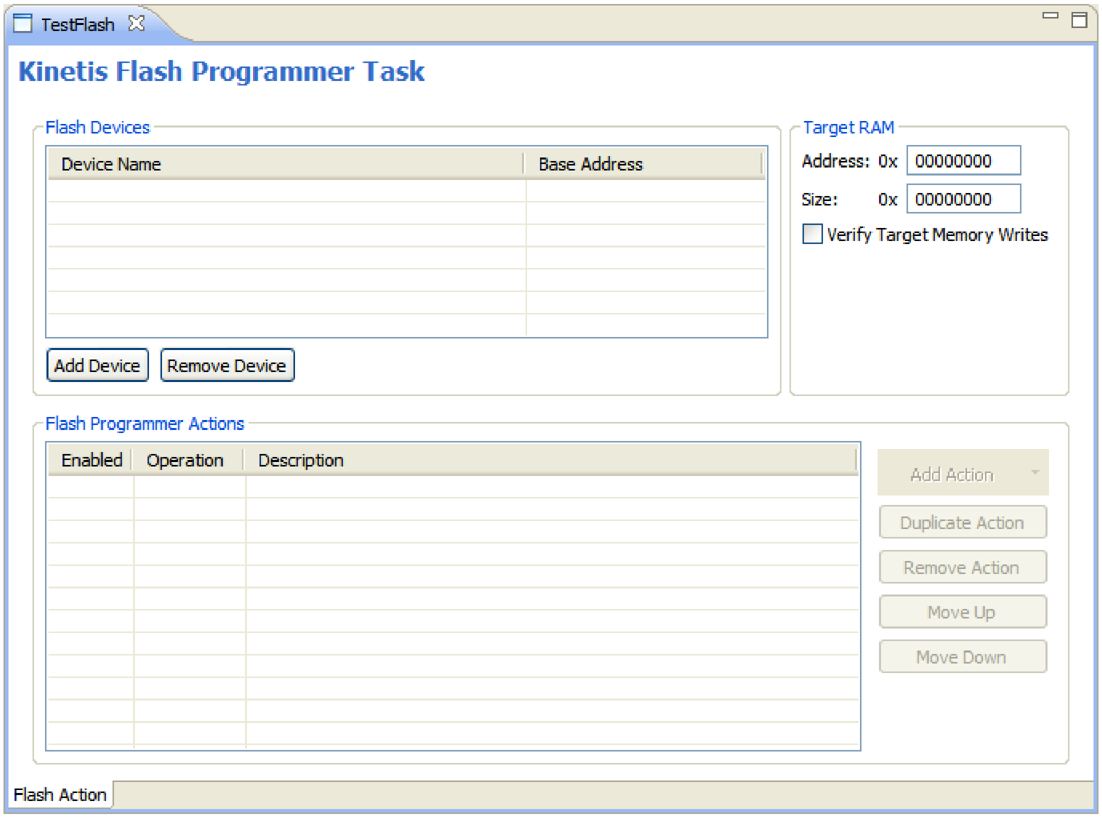
17.1.4 Configure the Flash Programmer Target Task

To configure a flash programmer target task, you need to perform the following actions.

- Adding Flash Device
- Specify Target RAM Settings
- Add Flash Programmer Actions

NOTE

Click the **Save** button or press **CTRL + S** to save the task settings.



Address: 0x 00000000

Size: 0x 00000000

□ Verify Target Memory Writes

Figure 17-6. Flash Programmer Task Editor Window

The table below lists the groups in the **Flash Programmer Task Editor** window.

Table 17-2. Flash Programmer Task Editor Window - Groups

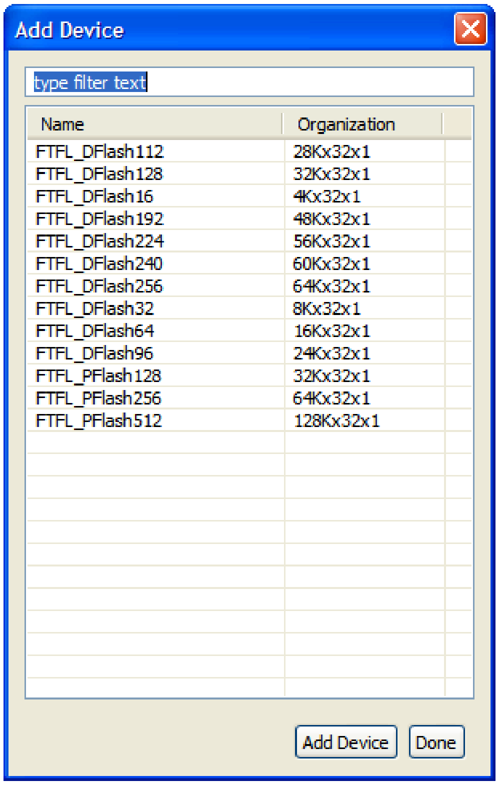
Group	Description
Flash Devices	Lists the devices added in the current task.
Target RAM	Enables you to specify the settings for Target RAM.
Flash Program Actions	Lists the programmer actions to be performed on the flash devices.

17.1.4.1 Adding Flash Device

To add a flash device to the **Flash Devices** table:

1. Click the **Add Device** button.

The **Add Device** dialog box appears.



type filter text

Name	Organization
FTFL_DFlash112	28Kx32x1
FTFL_DFlash128	32Kx32x1
FTFL_DFlash16	4Kx32x1
FTFL_DFlash192	48Kx32x1
FTFL_DFlash224	56Kx32x1
FTFL_DFlash240	60Kx32x1
FTFL_DFlash256	64Kx32x1
FTFL_DFlash32	8Kx32x1
FTFL_DFlash64	16Kx32x1
FTFL_DFlash96	24Kx32x1
FTFL_PFlash128	32Kx32x1
FTFL_PFlash256	64Kx32x1
FTFL_PFlash512	128Kx32x1

Figure 17-7. Add Device Dialog Box

2. Select a flash device from the device list.
3. Click **Add Device** from the **Flash Devices** group to add a new hardware device.

   The **Add Device** dialog box appears with a list of supported devices.

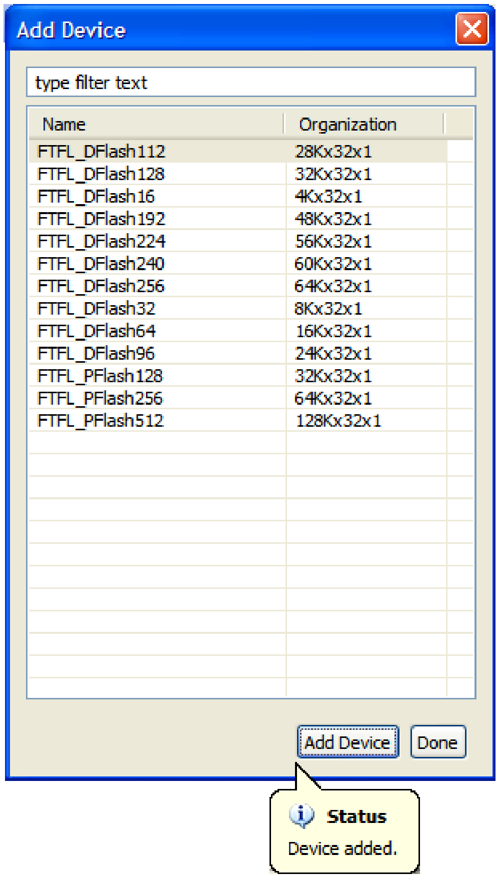
   a. Select the specific device from the list.
   b. Change the device's memory organization (if required).

   NOTE

   To change the device memory organization, click the adjacent value in the **Organization** column. Click the down button icon, and select the required organization.

   c. Click **Add Device**.

   You get a popup with a status that the device is added.



Name	Organization
FTFL_DFlash112	28Kx32x1
FTFL_DFlash128	32Kx32x1
FTFL_DFlash16	4Kx32x1
FTFL_DFlash192	48Kx32x1
FTFL_DFlash224	56Kx32x1
FTFL_DFlash240	60Kx32x1
FTFL_DFlash256	64Kx32x1
FTFL_DFlash32	8Kx32x1
FTFL_DFlash64	16Kx32x1
FTFL_DFlash96	24Kx32x1
FTFL_PFlash128	32Kx32x1
FTFL_PFlash256	64Kx32x1
FTFL_PFlash512	128Kx32x1

Status
Device added.

Figure 17-8. Add Device Dialog Box - Popup with Status

d. You can select other devices, if required.

4. Click **Done**.

   The **Add Device** dialog box closes. The devices appear in the **Flash Devices** group of the <target> **Flash Programmer Task** editor window.

17.1.4.2 Specify Target RAM Settings

The Target RAM is used by Flash Programmer to download its algorithms.

NOTE

The Target RAM memory area is not restored by flash programmer. If you are using flash programmer with Active Debug Context, it will impact your debug session.

1. Enter the first address from target memory used by the flash algorithm (running on the target) in the **Address** text box of the **Target RAM** group.

a. Enter the size of the memory that the flash algorithm is allowed to use in the **Size** text box.
b. Check the **Verify Target Memory Writes** checkbox to verify all write operations to the hardware RAM during Flash programming.

2. From the **Flash Programmer Actions** group, you can use any of the buttons listed in the table below to add various Flash Programmer actions.

Table 17-3. Task Actions and Sequence Organization

Button/Option	Usage
Erase / Blank Check	Lets you add erase or blank check actions for flash devices
Program / Verify	Lets you add program or verify flash actions for flash devices
Checksum	Lets you add checksum actions for flash devices
Diagnostics	Lets you add a diagnostics action to the actions table.
Dump Flash	Lets you dump a portion of the flash or the entire flash.
Protect/Unprotect	Lets you modify the protection of a sector.
Secure/Unsecure	Lets you add secure or unsecure actions.
Remove Action	Lets you remove a sector, a group of sector, or an entire device from the **Flash Programmer Actions** table, depending on the flash capabilities.
Move Up	Lets you move a selected flash action up in the **Flash Programmer Actions** table, so that it executes before other actions beneath it in the table.
Move Down	Lets you move a selected flash action down in the **Flash Programmer Actions** table, so that the action executes after other actions above of it in the table execute.

17.1.4.3 Add Flash Programmer Actions

In the **Flash Programmer Actions** group in the Flash Programmer Task editor window, add the flash programmer actions to be performed on the flash device. You can perform the following actions on a flash device.

- Add Erase/Blank Check Actions
- Add Program / Verify Actions
- Add Checksum Actions
- Add Diagnostics Actions
- Add Dump Flash Actions

- Add Protect/Unprotect Actions
- Add Secure/Unsecure Actions

NOTE

Actions can also be enabled or disabled using the **Enabled** column. The **Description** column contains the default description for the flash programmer actions. You can also edit the default description.

17.1.4.3.1 Add Erase/Blank Check Actions

The erase action enables you erase sectors from the flash device. You can also use the erase action to erase the entire flash memory without selecting sectors. The blank check action verifies if the specified areas have been erased from the flash device.

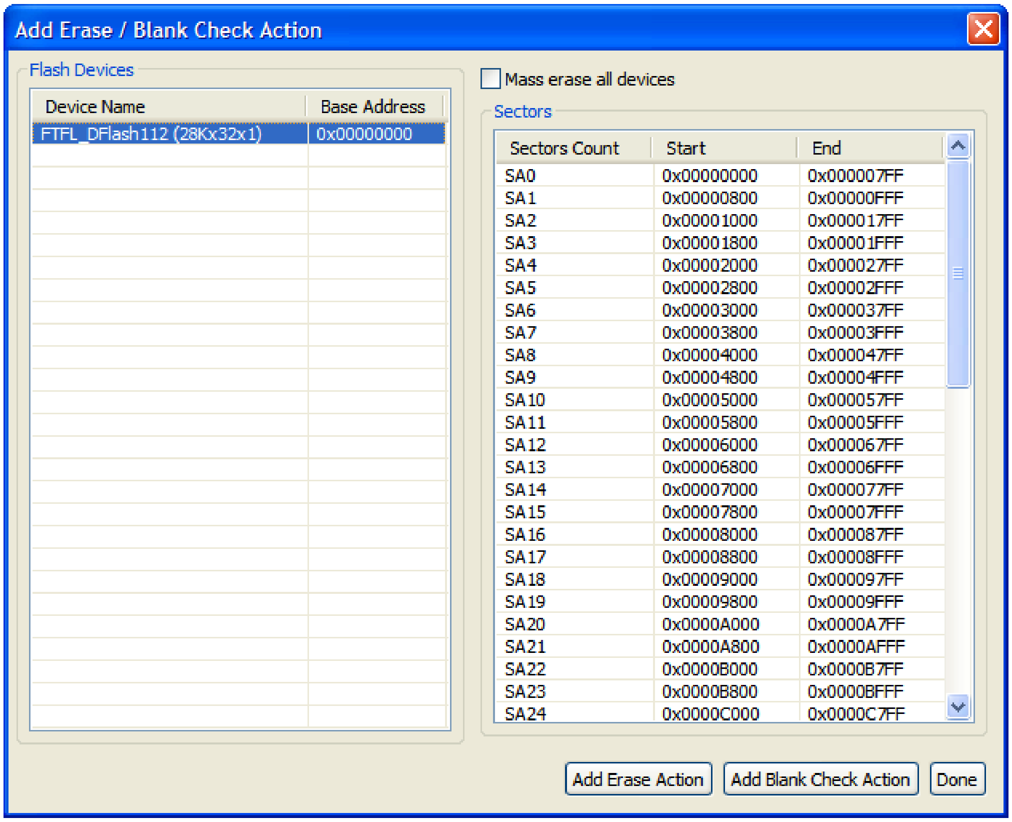
NOTE

Flash Programmer will not erase a bad sector in the NAND flash. After the erase action a list of bad sectors is reported (if any).

To add an erase/blank check action:

1. Select the **Erase / Blank Check** action from the **Add Action** drop-down list.

   The **Add Erase / Blank Check Action** dialog box appears.



Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

☐ Mass erase all devices

Sectors Count	Start	End
SA0	0x00000000	0x000007FF
SA1	0x00000800	0x00000FFF
SA2	0x00001000	0x000017FF
SA3	0x00001800	0x00001FFF
SA4	0x00002000	0x000027FF
SA5	0x00002800	0x00002FFF
SA6	0x00003000	0x000037FF
SA7	0x00003800	0x00003FFF
SA8	0x00004000	0x000047FF
SA9	0x00004800	0x00004FFF
SA10	0x00005000	0x000057FF
SA11	0x00005800	0x00005FFF
SA12	0x00006000	0x000067FF
SA13	0x00006800	0x00006FFF
SA14	0x00007000	0x000077FF
SA15	0x00007800	0x00007FFF
SA16	0x00008000	0x000087FF
SA17	0x00008800	0x00008FFF
SA18	0x00009000	0x000097FF
SA19	0x00009800	0x00009FFF
SA20	0x0000A000	0x0000A7FF
SA21	0x0000A800	0x0000AFFF
SA22	0x0000B000	0x0000B7FF
SA23	0x0000B800	0x0000BFFF
SA24	0x0000C000	0x0000C7FF

Figure 17-9. Add Erase / Blank Check Action Dialog Box

2. Select a sector from the **Sectors** table and click the **Add Erase Action** button to add an erase operation on the selected sector.

NOTE

Press **CTRL** or **SHIFT** keys for selecting multiple sectors from the **Sectors** table.

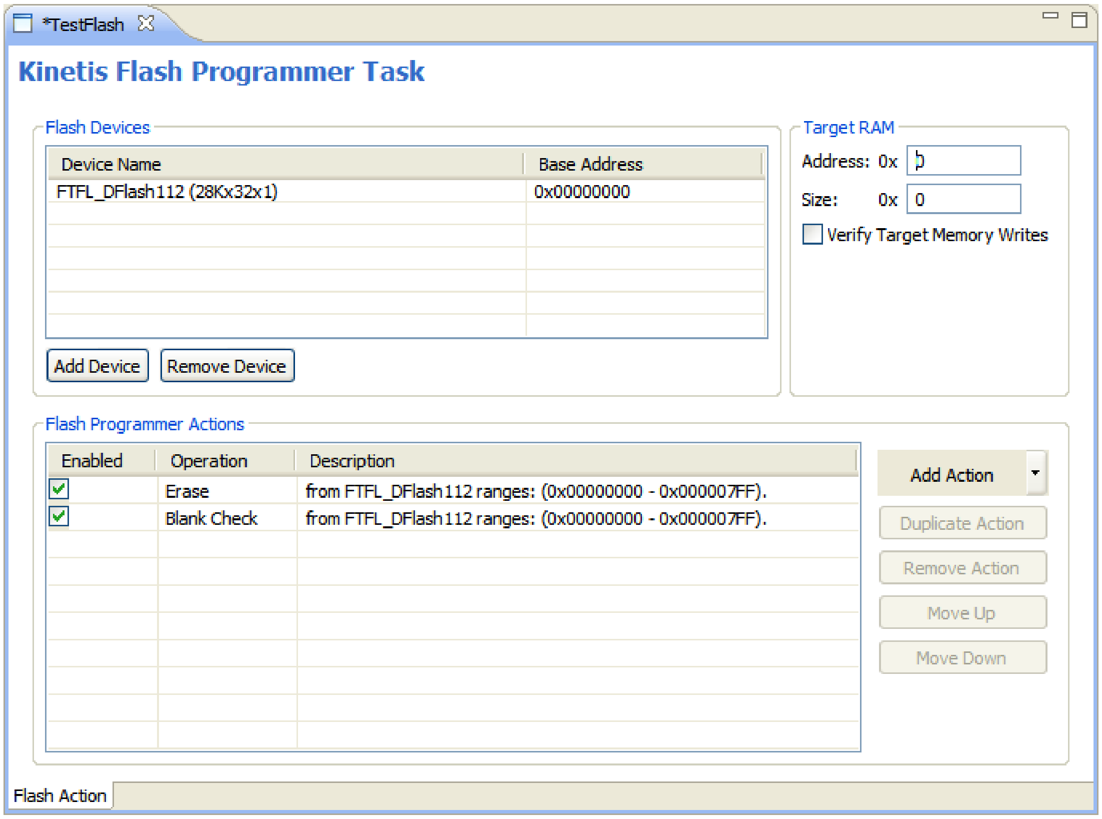
3. Click the **Add Blank Check Action** button to add a blank check operation on the selected sector.
4. Check the **Mass erase all devices** check box to erase the entire flash memory for HCS08/RS08/CFv1/Kinetis.

NOTE

After checking the **Erase All Sectors Using Chip Erase Command** check box (for CFv2, DSC, Power Architecture), you need to add either erase or blank check action to erase all sectors. For S12Z use the **Erase entry flash block**.

5. Click **Done**.

The **Add Erase / Blank Check Action** dialog box closes and the added erase / blank check actions appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.



Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

Address: 0x

Size: 0x 0

□ Verify Target Memory Writes

Enabled	Operation	Description
☑	Erase	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Blank Check	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).

Figure 17-10. Added Erase / Blank Check Actions

17.1.4.3.2 Add Program/Verify Actions

The program action enables you to program the flash device and the verify action verifies the programmed flash device.

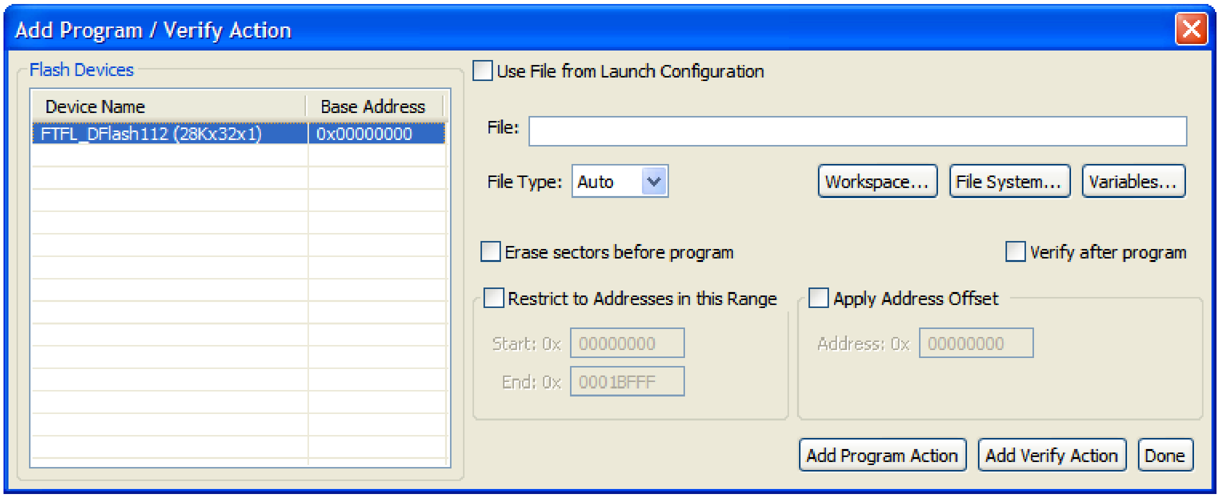
NOTE

The program action will abort and fail if it is performed in a bad block for NAND flashes.

To add a program/verify action:

1. Select the **Program / Verify** action from the **Add Action** drop-down list.

   The **Add Program / Verify** dialog box appears.



Add Program / Verify Action

Flash Devices

Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

☐ Use File from Launch Configuration

File:

File Type: Auto

☐ Erase sectors before program

☐ Verify after program

☐ Restrict to Addresses in this Range

Start: 0x 00000000

End: 0x 0001BFFF

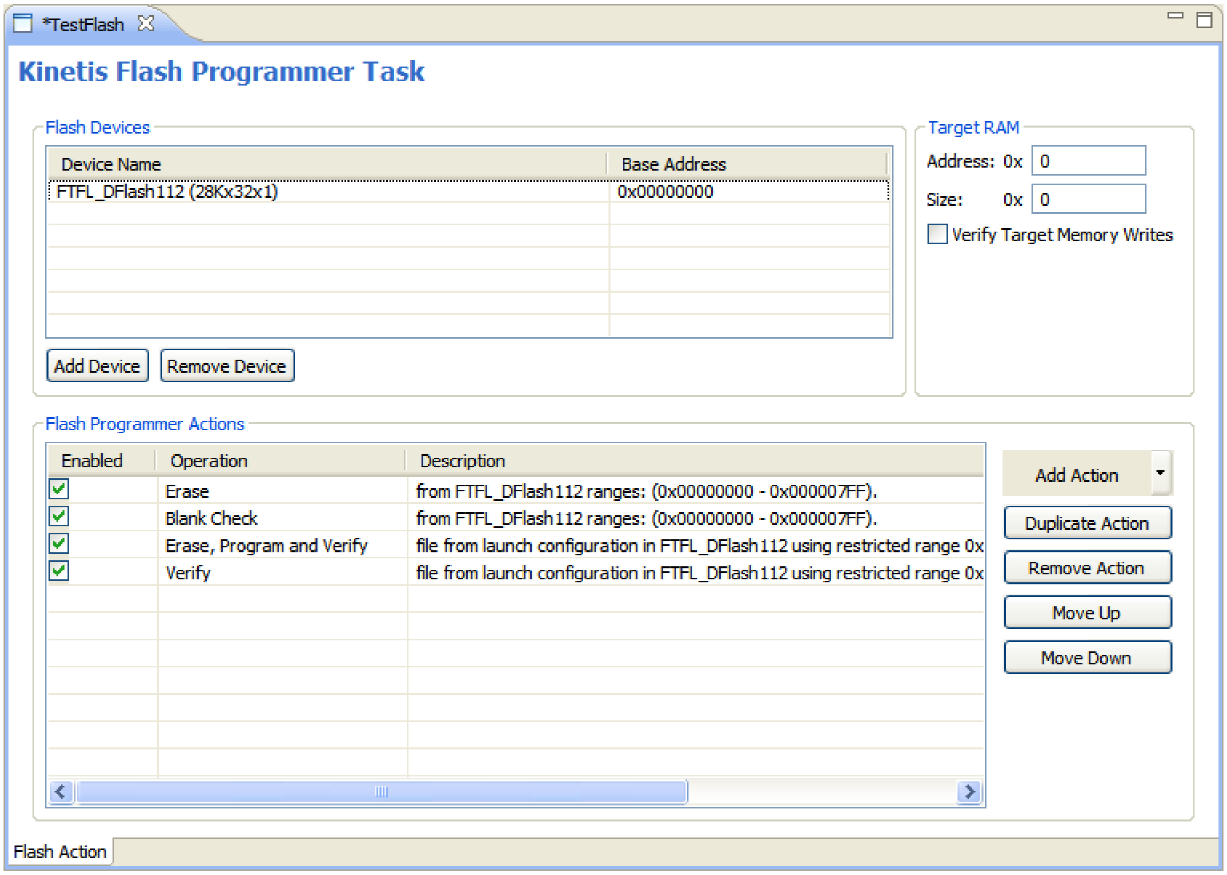
☐ Apply Address Offset

Address: 0x 00000000

Figure 17-11. Add Program / Verify Action Dialog Box

2. Select the file to be written to the flash device.
   - Check the **Use File from Launch Configuration** check box to use the file from the launch (run) configuration associated with the task.
   - Specify the file name in the **File** text box. You can use **Workspace**, **File System**, or **Variables** buttons to select the desired file.
3. Select the file type from the **File Type** drop-down list. You can select any one of the following file types:
   - Auto - Detects the file type automatically.
   - Abs/Elf - Specifies executable in ABS or ELF format.
   - Srec - Specifies files in Motorola S-record format.
   - Binary - Specifies binary files.
4. Check the **Erase sectors before program** checkbox to erase sectors before program.
5. Check the **Verify after program** checkbox to verify after the program.
6. Check the **Restricted To Addressin the Range** check box to specify a memory range. The write action is permitted only in the specified address range. In the **Start** text box, specify the start address of the memory range sector and in the **End** text box, specify the end address of the memory range.
7. Check the **Apply Address Offset** check box and set the memory address in the **Address** text box. Value is added to the start address of the file to be programmed or verified.
8. Click the **Add Program Action** button to add a program action on the flash device.
9. Click the **Add Verify Action** button to add a verify action on the flash device.
10. Click **Done**.

    The **Add Program/Verify Action** dialog box closes and the added program / verify actions appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.



Kinetis Flash Programmer Task

Flash Devices

Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

Target RAM

Address: 0x 0

Size: 0x 0

□ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Blank Check	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Erase, Program and Verify	file from launch configuration in FTFL_DFlash112 using restricted range 0x
☑	Verify	file from launch configuration in FTFL_DFlash112 using restricted range 0x

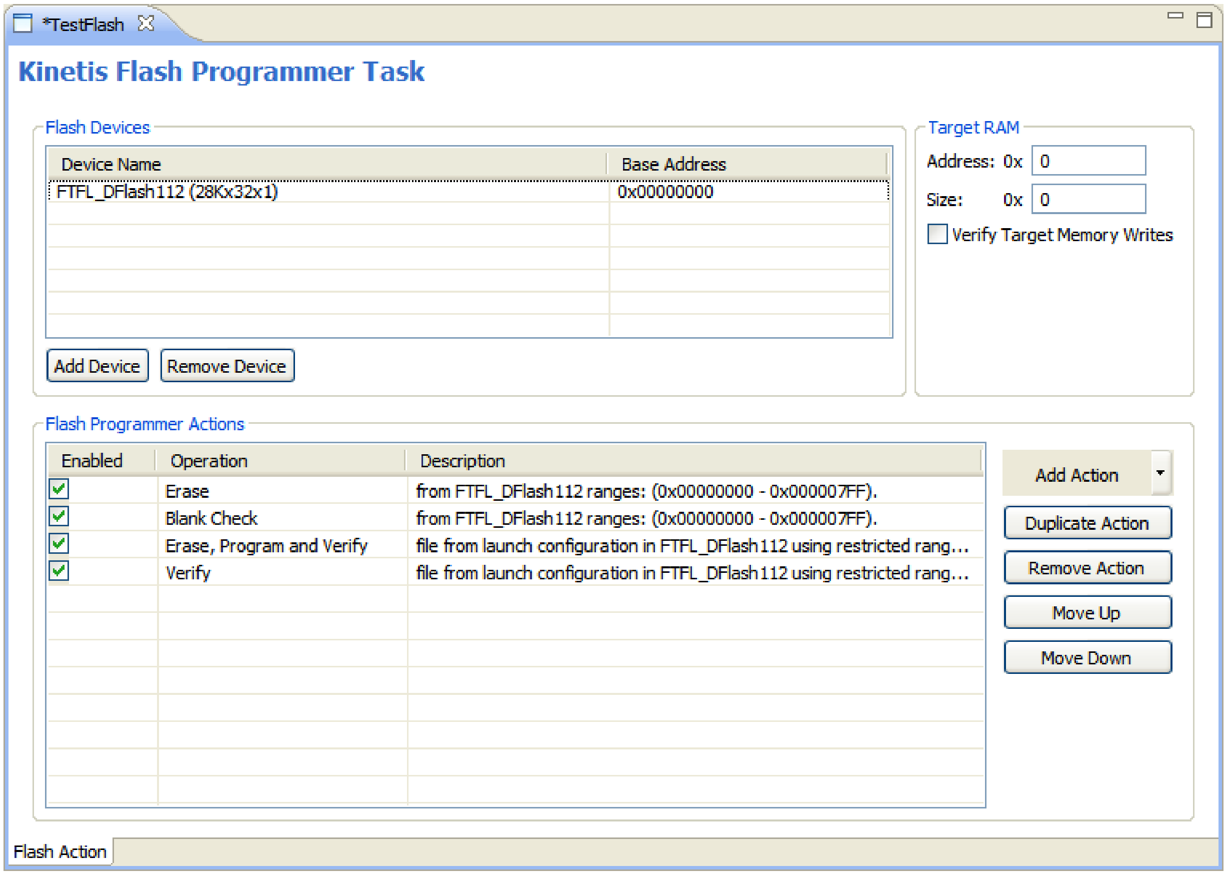
Figure 17-12. Added Program/Verify Actions

17.1.4.3.3 Add Checksum Actions

The checksum can be computed over host file, target file, memory range or entire flash memory. To add a checksum action:

1. Select the **Checksum** action from the **Add Action** drop-down list.

   The **Add Checksum** dialog box appears.



*TestFlash

Kinetis Flash Programmer Task

Flash Devices

Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

Target RAM

Address: 0x 0

Size: 0x 0

□ Verify Target Memory Writes

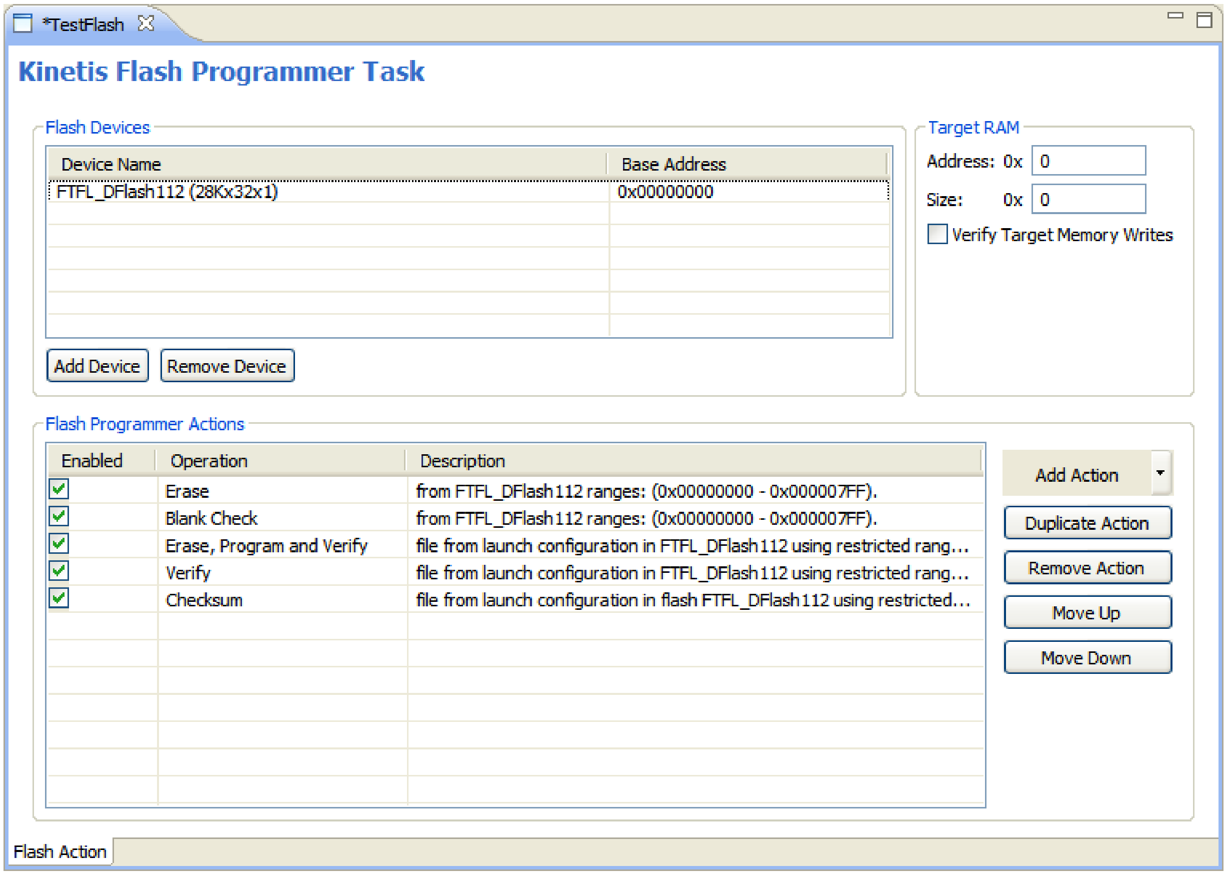
Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Blank Check	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Erase, Program and Verify	file from launch configuration in FTFL_DFlash112 using restricted rang...
☑	Verify	file from launch configuration in FTFL_DFlash112 using restricted rang...

Figure 17-13. Add CheckSum Action Dialog Box

2. Select the file for checksum action.
    - Check the **Use File from Launch Configuration** check box to use the file from the launch (run) configuration associated with the task.
    - Specify the filename in the **File** text box. You can use the **Workspace**, **File System**, or **Variables** buttons to select the desired file.
3. Select the file type from the **File Type** drop-down list.
4. Select an option from the **Compute Checksum Over** options. The checksum can be computed over the host file, the target file, the memory range, or the entire flash memory.
5. Specify the memory range in the **Restricted To Addresses in the Range** group. The checksum action is permitted only in the specified address range. In the **Start** text box, specify the start address of the memory range sector and in the **End** text box, specify the end address of the memory range.
6. Check the **Apply Address Offset** check box and set the memory address in the **Address** text box. Value is added to the start address of the file to be programmed or verified.
7. Click the **Add Checksum Action** button.
8. Click **Done.**

The **Add Checksum Action** dialog box closes and the added checksum actions appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.



Kinetis Flash Programmer Task

Flash Devices

Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

Target RAM

Address: 0x 0

Size: 0x 0

☐ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Blank Check	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Erase, Program and Verify	file from launch configuration in FTFL_DFlash112 using restricted rang...
☑	Verify	file from launch configuration in FTFL_DFlash112 using restricted rang...
☑	Checksum	file from launch configuration in flash FTFL_DFlash112 using restricted...

Figure 17-14. Added Checksum Actions

17.1.4.3.4 Add Diagnostics Actions

The diagnostics action generates the diagnostic information for a selected flash device.

NOTE

Flash Programmer will report bad blocks, if they are present in the NAND flash.

To add a diagnostics action:

1. Select the **Diagnostics** action from the **Add Action** drop-down list.

   The **Add Diagnostics** dialog box appears.

2. Select a device to perform the diagnostics action.

3. Click the **Add Diagnostics Action** button to add diagnostic action on the selected flash device.

   NOTE

   Check the **Perform Full Diagnostics** check box to perform full diagnostics on a flash device.

4. Click **Done**.

   The **Add Diagnostics Action** dialog box closes and the added diagnostics action appears in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.

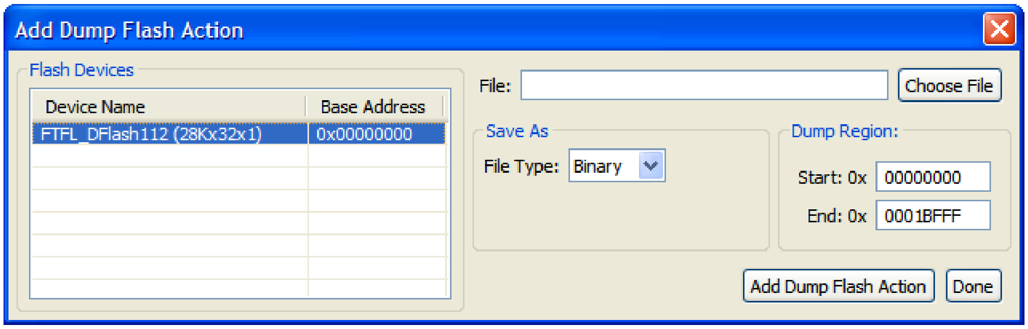
17.1.4.3.5 Add Dump Flash Actions

The dump flash action enables you to dump selected sectors of a flash device or the entire flash device.

To add a dump flash action:

1. Select the **Dump Flash** action from the **Add Action** drop-down list.

   The **AddDump Flash Action** dialog box appears.



Add Dump Flash Action

Flash Devices

Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

File:

Save As

File Type: Binary

Dump Region:

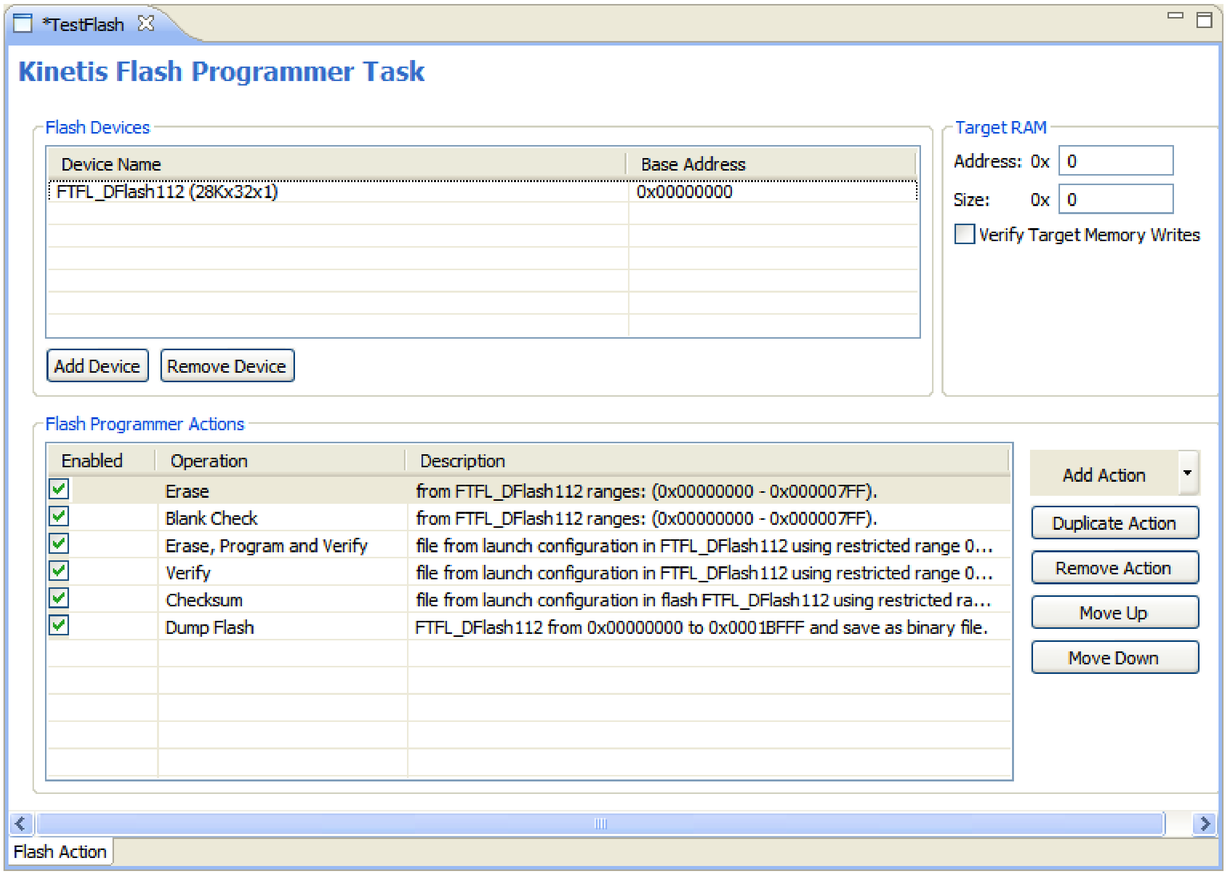
Start: 0x 00000000

End: 0x 0001BFFF

Figure 17-15. Add Dump Flash Action Dialog Box

2. Specify the file name in the **File** text box. The flash is dumped in this selected file.
3. Select the file type from the **File Type** drop-down list. You can select any one of the following file types:
   - Srec - Saves files in Motorola S-record format.
   - Binary - Saves files in binary file format.
4. Specify the memory range for which you want to add dump flash action.
   - Type the start address of the range in the **Start** text box.
   - Type the end address of the range in the **End** text box.
5. Click the **Add Dump Flash Action** button to add a dump flash action.
6. Click **Done**.

The **Add Dump Flash Action** dialog box closes and the added dump flash action appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.



Kinetis Flash Programmer Task

Flash Devices

Device Name	Base Address
FTFL_DFlash112 (28Kx32x1)	0x00000000

Target RAM

Address: 0x 0

Size: 0x 0

□ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Blank Check	from FTFL_DFlash112 ranges: (0x00000000 - 0x000007FF).
☑	Erase, Program and Verify	file from launch configuration in FTFL_DFlash112 using restricted range 0...
☑	Verify	file from launch configuration in FTFL_DFlash112 using restricted range 0...
☑	Checksum	file from launch configuration in flash FTFL_DFlash112 using restricted ra...
☑	Dump Flash	FTFL_DFlash112 from 0x00000000 to 0x0001BFFF and save as binary file.

Figure 17-16. Added Dump Flash Actions

17.1.4.3.6 Add Protect/Unprotect Actions

The protect/unprotect actions enable you to change the protection of a sector in the flash device.

To add a protect/unprotect action:

1. Select the **Protect/Unprotect** action from the **Add Action** drop-down list.

   The **Add Protect/Unprotect Action** dialog box appears.

2. Select a sector from the **Sectors** table and click the **Add Protect Action** button to add a protect operation on the selected sector.

NOTE

Press **CTRL** or **SHIFT** keys for selecting multiple sectors from the **Sectors** table.

3. Click the **Add Unprotect Action** button to add an unprotect action on the selected sector.
4. Check the **All Device** check box to add action on full device.
5. Click **Done**.

   The **Add Protect/Unprotect Action** dialog box closes and the added protect or unprotect actions appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.

17.1.4.3.7 Add Secure/Unsecure Actions

The secure/unsecure actions enable you to change the security of a flash device.

To add a secure/unsecure action:

1. Select the **Secure/Unsecure** action from the **Add Action** drop-down list.

   The **Add Secure/UnSecure Action** dialog box appears.

2. Select a device from the **Flash Devices** table.
3. Click the **Add Secure Action** button to add Secure action on the selected flash device.
   a. Type a password in the **Password** text box.
   b. Select the password format from the **Format** drop-down list box.
4. Click the **Add Unsecure Action** button to add an unprotect action on the selected sector.
5. Click **Done**.

   The **Add Secure/UnSecure Action** dialog box closes and the added secure or unsecure action appears in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.

17.1.4.3.8 Remove an Action

To remove a flash programmer action from the **Flash Programmer Actions** table:

1. Select the action in the **Flash Programmer Actions** table.
2. Click the **Remove Action** button.

The selected action is removed from the **Flash Programmer Action** table.

17.1.5 Run Flash Programmer Target Task

To execute the configured flash programmer target task, select a target task and click the **Execute** button in the **Target Tasks** view toolbar. Alternatively, right-click on a target task and select **Execute** from the context menu.

NOTE

> You can use predefined target tasks for supported boards. To load a predefined target task, right-click in the Target Tasks view and select **Import Target Task** from the context menu. To save your custom tasks, right-click in the Target Tasks view and then select **Export Target Task** from the context menu.

You can check the results of flash batch actions in the **Console** view. The green color indicates the success and the red color indicates the failure of the task.

17.1.6 Create Flash Programmer Task

In the Eclipse IDE, the Flash Programmer runs like a target task.

To create a Flash Programmer target task:

1. From the CodeWarrior main menu bar, select **Window > Show View > Other**.

   The **Show View** dialog box appears.

2. Expand the **Debug** tree control and select **Target Tasks**.

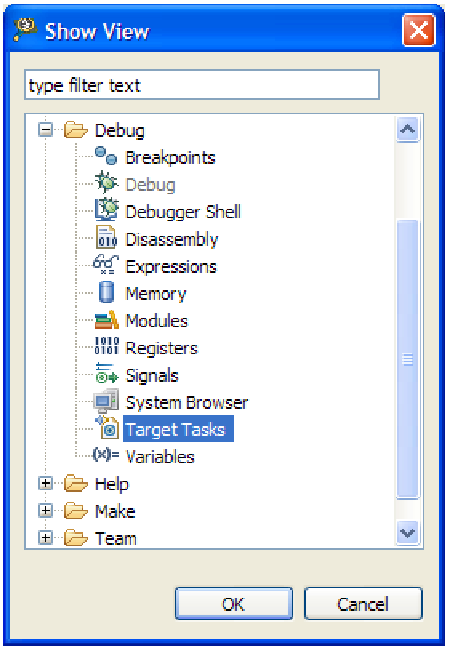


Figure 17-17. Show View Dialog Box

3. Click **OK**.

   The **Target Tasks** view appears.

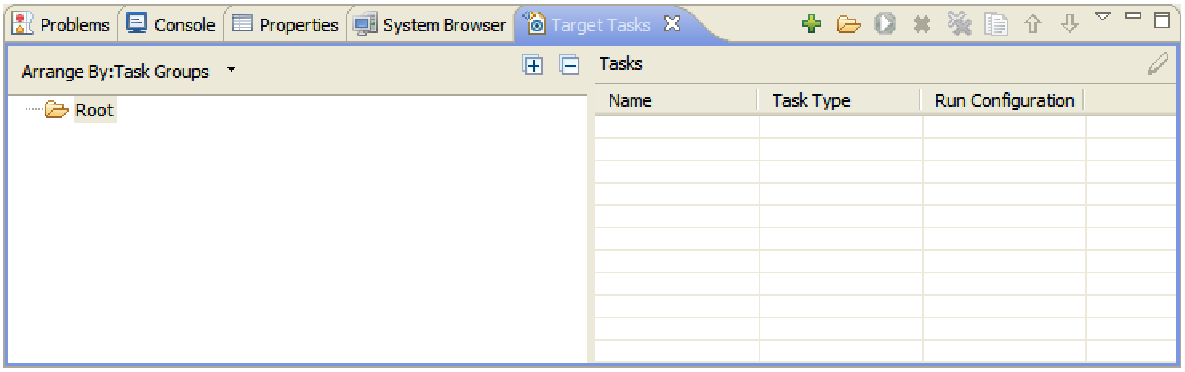


Figure 17-18. Target Tasks View

4. Click the **add** icon from the **Target Tasks** view toolbar to create a new target task.

   The **Create New Target Task** wizard appears.

17.1.7 Flash Programmer for MCF5441X

The flash programmer for MCF5441X has a specific way of writing data. The flash is comprised of multiple physical pages, each of them with the size of 0x800 bytes. At boot time the processor will read data from the first 4 pages up to 0xF80 bytes. The space between address 0xF80 and 0x2000 will not be used. Assuming we write a binary file, the data will be programmed as follows:

- Binary file address 0x0 - 0xF7F will be programmed at 0x0 - 0xF7f in flash.
- Binary file addresses higher than 0xF80 will be programmed starting from 5th physical page at 0x2000.

Because of the way how the processor boots there isn't a 1 to 1 correspondence for addresses. The pages are written with the following settings:

- Boot pages (index 0 - 3): 60 ECC bytes, in NFC_CFG register BTMD is 1 and 16BIT is 0.
- User pages (index > 3): 60 ECC bytes, in NFC_CFG register BTMD is 0 and 16BIT is 1.

17.2 Flash Programmer Tutorials

This topic consists of two tutorials that demonstrate how to import and execute pre-defined target tasks. Additionally, there are tutorials on how use the <target> **Flash Programmer Task** editor window to create tasks, such as erasing on-chip memory, or downloading a file, and writing it into flash memory.

NOTE

The Flash Programmer for RS08 works only if OSBDM connection is used.

The tutorials include:

- Tutorial A: Import and Execute HCS08 Flash Task
- Tutorial B: Import and Execute ColdFire Flash Task
- Tutorial C: Create Erase Memory Task for HCS08
- Tutorial D: Create Erase Flash Memory Task for ColdFire
- Tutorial E: Create Download Program Task for ColdFire
- Tutorial F: Import and Execute Power Architecture Flash Task
- Tutorial G: Switching Between Lock-Step and Decoupled Parallel Modes
- Tutorial H: Create and Execute Diagnostics Action Task
- Tutorial I: Dump Entire Flash
- Tutorial J: Change Protection of Sector
- Tutorial K: Fast Access to Target Tasks Editors

- Tutorial L: Programming with Simple Flash
- Tutorial M: Exporting Target Tasks

17.2.1 Tutorial A: Import and Execute HCS08 Flash Task

The goal of this tutorial is to select and import a pre-defined task that erases and programs the flash memory in a DEMOS908QG8, an evaluation board based on a Freescale MC9S08QG8 microcontroller.

- Import HCS08 Program Flash Task
- Execute MC9S08QG8 Task

17.2.1.1 Import HCS08 Program Flash Task

NOTE

> This procedure assumes that the **Target Tasks** view is visible in the perspective. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

After you have launched CodeWarrior and connected the DEMOS908QG8 board to the workstation using a USB cable:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **Import**. Alternatively, click the **Import** icon on the **Target Tasks** view toolbar.

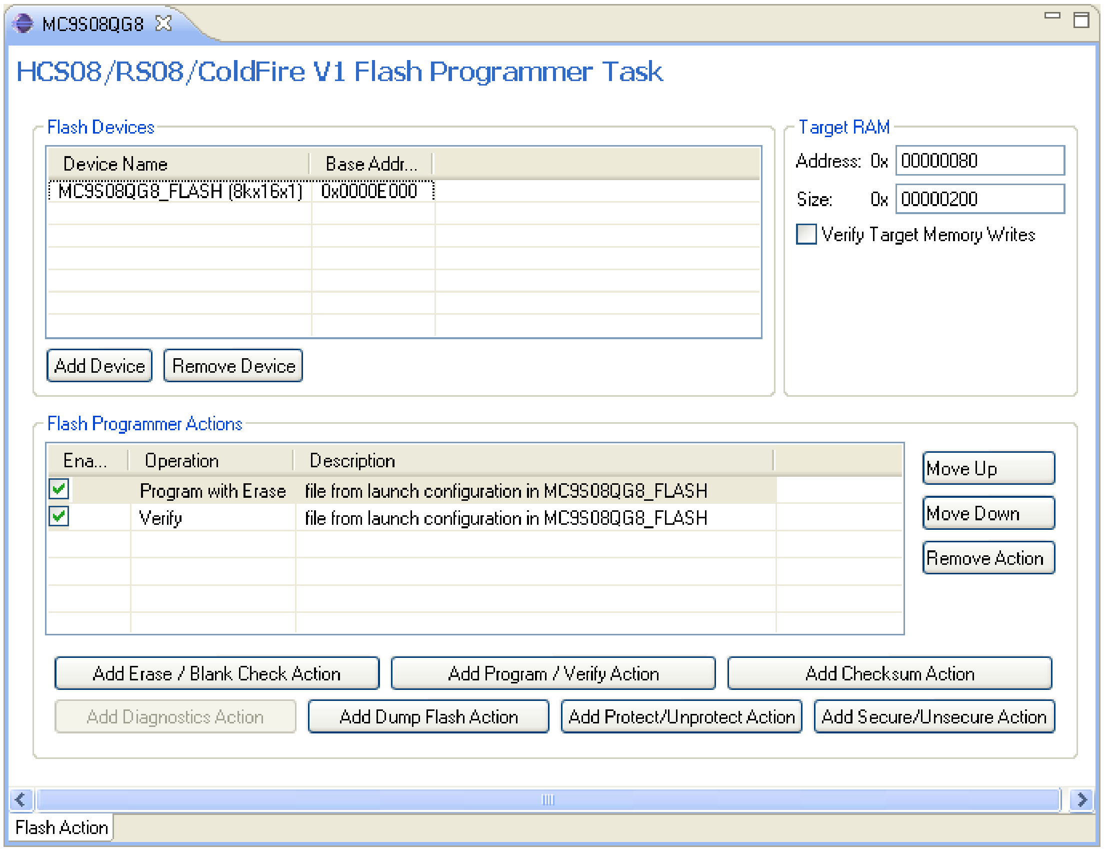
   The **Open** dialog box appears.

3. Navigate to `<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/HC08` and select the XML file for the board's microcontroller. For this tutorial, select `MC9S08QG8.xml`.
4. Click **Open**.

   The MC9S08QG8 task appears in the **Target Tasks** view.

5. Double-click on the **MC9S08QG8** task to examine its contents.

The <target> **Flash Programmer Task** editor window appears, and displays the memory settings and actions for the task. Notice the actions to erase, program, and verify the contents of flash memory in the **Flash Programmer Actions** group. These actions execute in the order as they are displayed in the table, from top to bottom.



HCS08/RS08/ColdFire V1 Flash Programmer Task

Flash Devices

Device Name	Base Addr...
MC9S08QG8_FLASH (8kx16x1)	0x0000E000

Target RAM

Address: 0x 00000080

Size: 0x 00000200

☐ Verify Target Memory Writes

Flash Programmer Actions

Ena...	Operation	Description
☑	Program with Erase	file from launch configuration in MC9S08QG8_FLASH
☑	Verify	file from launch configuration in MC9S08QG8_FLASH

Figure 17-19. Memory Settings and Actions for MC9S08QG8 Task

17.2.1.2 Execute MC9S08QG8 Task

To execute the task:

1. Right-click on the MC9S08QG8 task.
2. Select **Execute**. Alternatively, select the MC9S08QG8 task and click the **Execute** icon on the **Target Tasks** view toolbar.

   CodeWarrior establishes contact with the DEMOS908QG8 board, erases the microcontroller's flash memory, downloads the code, and verifies that the contents of flash match those of the file.

Congratulations! You have selected and used a target task that erased and programmed the flash memory on the MC9S08QG8 microcontroller.

17.2.2 Tutorial B: Import and Execute ColdFire Flash Task

The goal of this tutorial is to select and import a pre-defined task that erases and then programs the flash memory in a board based on the ColdFire MCF5213 microcontroller.

- Import MCF5213 Program Flash Task
- Execute MCF5213_INTFLASH Task

17.2.2.1 Import MCF5213 Program Flash Task

> **NOTE**
>
> This tutorial assumes that you have already built a ColdFire project with the name **ColdFire_test**. It also assumes that you have previously created other tasks, so that the **Target Tasks** view is visible. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

After you have connected an MCF5213-based evaluation board to the workstation with a USB cable:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **Import** . Alternatively, click the **Export** icon on the **Target Tasks** view toolbar.

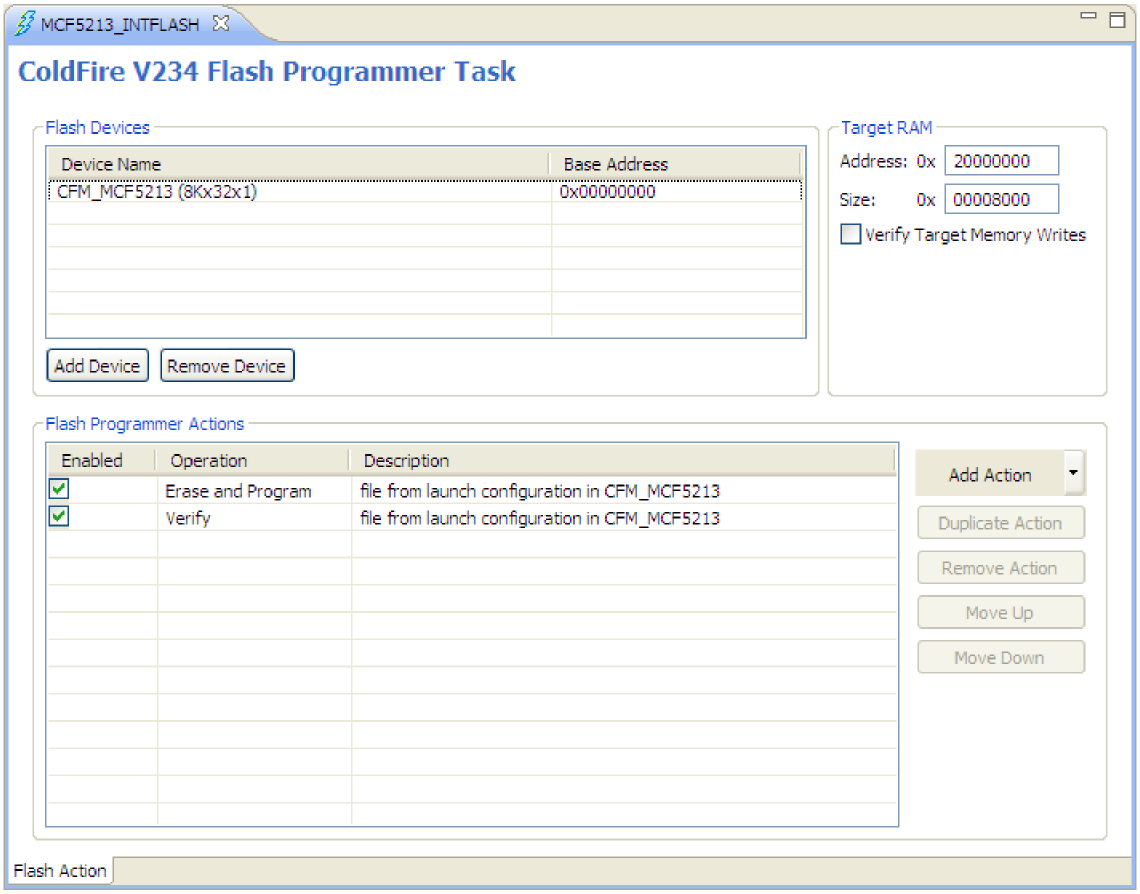
   The **Open** dialog box appears.

3. Navigate to `<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/ColdFire` and select the XML file for the board's microcontroller. For this tutorial, select `MCF5213_INTFLASH.xml`.
4. Click **Open** .

   The `MCF5213_INTFLASH` task appears in the **Target Tasks** view.

5. Double-click on the `MCF5213_INTFLASH` task to examine its contents.

The **<target> Flash Programmer Task** editor window appears, and displays the memory settings and actions for the task. Notice the actions to erase, program, and verify the contents of flash memory in the **Flash Programmer Actions** group. These actions execute in the order as they are displayed in the table, from top to bottom.



ColdFire V234 Flash Programmer Task

Flash Devices

Device Name	Base Address
CFM_MCF5213 (8Kx32x1)	0x00000000

Target RAM

Address: 0x 20000000

Size: 0x 00008000

□ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase and Program	file from launch configuration in CFM_MCF5213
☑	Verify	file from launch configuration in CFM_MCF5213

Figure 17-20. Memory Settings and Actions for the MCF5213_INTFLASH Task

17.2.2.2 Execute MCF5213_INTFLASH Task

To execute the task:

1. Right-click on the `MCF5213_INTFLASH` task.
2. Select **Execute** . Alternatively, select the `MCF5213_INTFLASH` task and click the **Execute** icon on the **Target Tasks** view toolbar.

   CodeWarrior establishes contact with the MCF5213-based board board, erases the microcontroller's flash memory, downloads the code, and verifies that the contents of flash match those of the file.

Congratulations! You have selected and used a target task that erased and programmed the flash memory on the MCF5213 microcontroller.

17.2.3 Tutorial C: Create Erase Memory Task for HCS08

The goal of this tutorial is to demonstrate how to create a task that erases the flash memory in a DEMOS908QG8, an evaluation board based on a Freescale MC9S08QG8 microcontroller.

- Set Up HCS08 Erase Task
- Execute HCS08 Erase Task

17.2.3.1 Set Up HCS08 Erase Task

NOTE

This procedure assumes that you have already created a project named `DEMO9S08QG8_test`. It also assumes that you have previously created other tasks, so that the **Target Tasks** view is visible. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

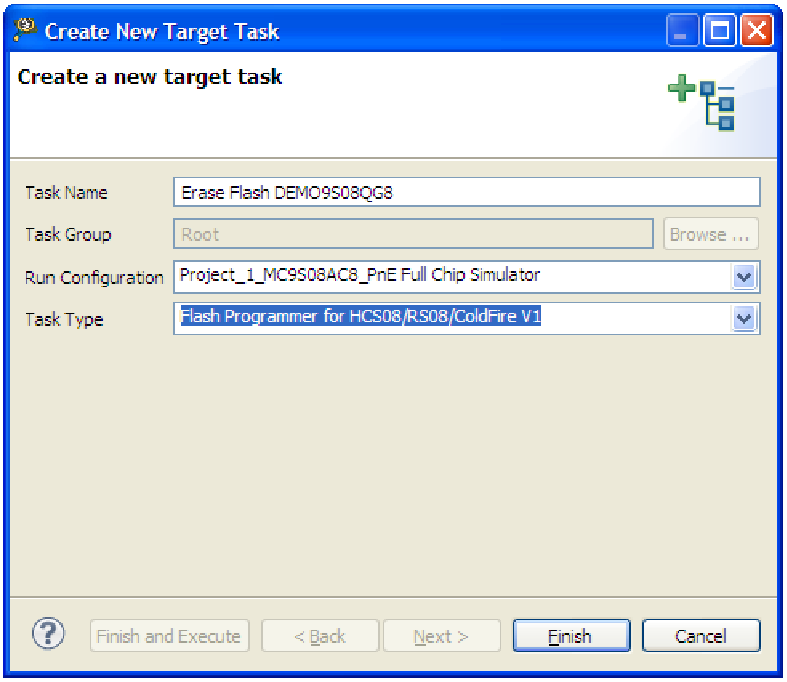
After you have launched CodeWarrior and connected the DEMOS908QG8 board to the host system using a USB cable:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **New Task**.

   The **Create New Target Task** wizard appears.

3. In the **Task Name** text box, enter the name of the target task. For example, enter `Erase Flash DEMO9S08QG8`.
4. From the **Run Configuration** drop-down list, select a run configuration.
5. From the **Task Type** drop-down list, select Flash Programmer for HCS08/RS08/ ColdFire V1 from the listbox.

   The dialog box should appear.



Create a new target task

Task Name: Erase Flash DEMO9S08QG8

Task Group: Root

Browse ...

Run Configuration: Project_1_MC9S08AC8_PnE Full Chip Simulator

Task Type: Flash Programmer for HCS08/RS08/ColdFire V1

Figure 17-21. Task Settings for Erasing Flash on MC9S08QG8

6. Click **Finish** to create the task.

   The <target> **Flash Programmer Task** editor window appears.

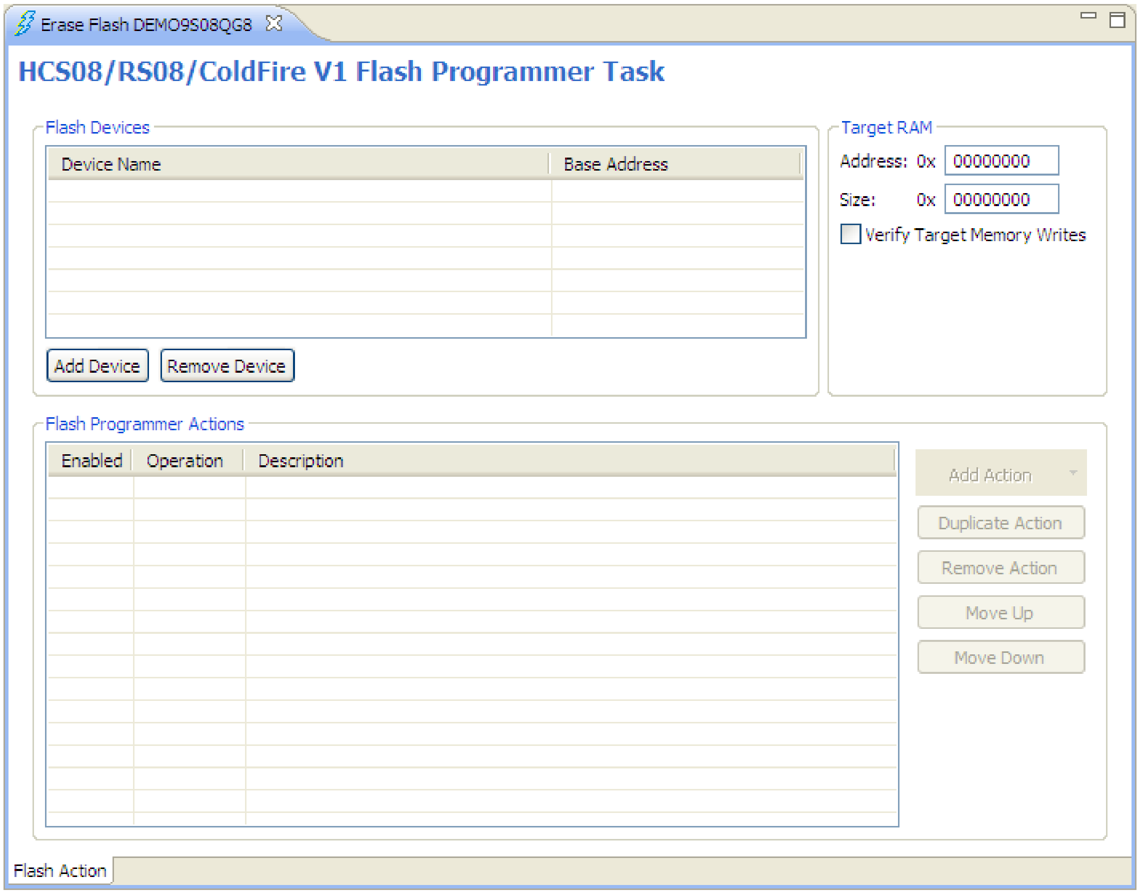
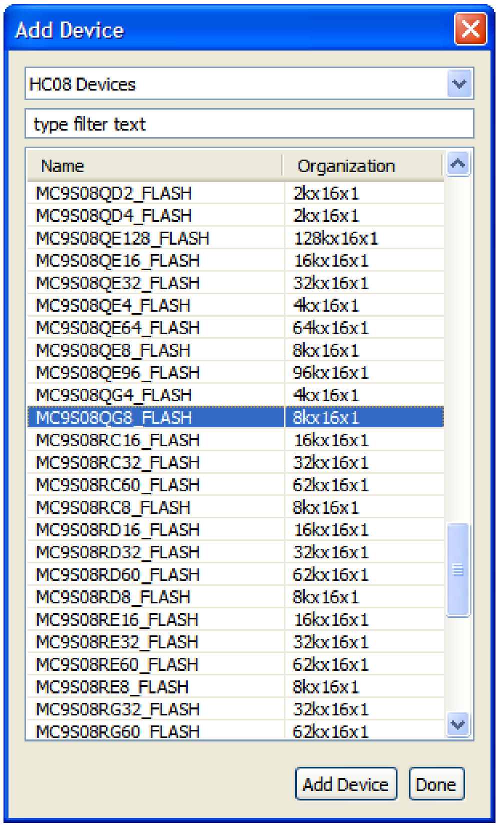


Figure 17-22. <target> Flash Programmer Task Editor Window to Erase HCS08 Flash

7. Click **Add Device**.

   The **Add Device** dialog box appears.



Name	Organization
MC9S08QD2_FLASH	2kx16x1
MC9S08QD4_FLASH	2kx16x1
MC9S08QE128_FLASH	128kx16x1
MC9S08QE16_FLASH	16kx16x1
MC9S08QE32_FLASH	32kx16x1
MC9S08QE4_FLASH	4kx16x1
MC9S08QE64_FLASH	64kx16x1
MC9S08QE8_FLASH	8kx16x1
MC9S08QE96_FLASH	96kx16x1
MC9S08QG4_FLASH	4kx16x1
MC9S08QG8_FLASH	8kx16x1
MC9S08RC16_FLASH	16kx16x1
MC9S08RC32_FLASH	32kx16x1
MC9S08RC60_FLASH	62kx16x1
MC9S08RC8_FLASH	8kx16x1
MC9S08RD16_FLASH	16kx16x1
MC9S08RD32_FLASH	32kx16x1
MC9S08RD60_FLASH	62kx16x1
MC9S08RD8_FLASH	8kx16x1
MC9S08RE16_FLASH	16kx16x1
MC9S08RE32_FLASH	32kx16x1
MC9S08RE60_FLASH	62kx16x1
MC9S08RE8_FLASH	8kx16x1
MC9S08RG32_FLASH	32kx16x1
MC9S08RG60_FLASH	62kx16x1

Figure 17-23. Add Device for HCS08 Derivative

8. Scroll through the list of microcontroller derivatives and select MC9S08QG8_FLASH. You can also type the filter text in the given text box.
9. Click **Add Device**.
10. Click **Done**.

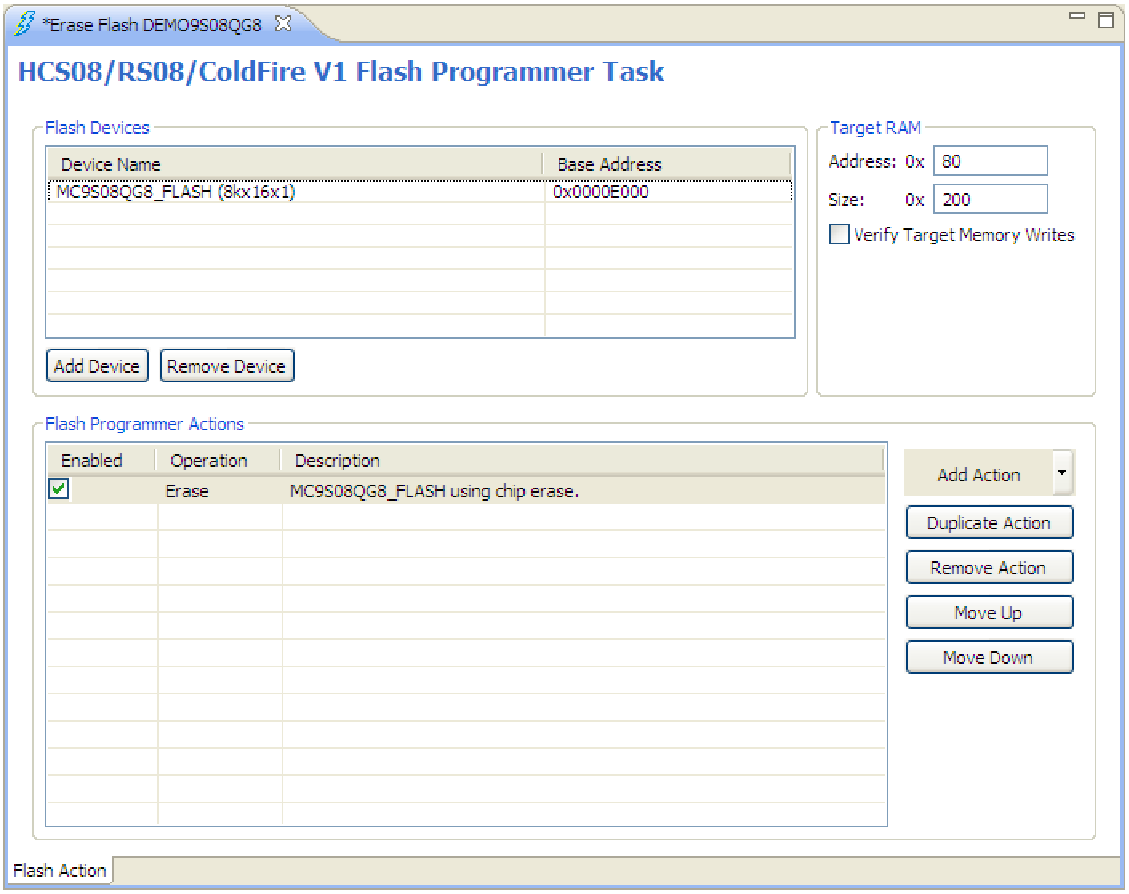
    You return to the <target> **Flash Programmer Task** editor window, with the selected device appearing in the **Flash Devices** group. The **Target RAM** group displays the start address of RAM memory, and its size.

11. Select the **Erase / Blank Check** action from the **Add Action** drop-down list.

    The **Add Erase / Blank Check Action** dialog box appears. It displays the flash devices added in the task and their base addresses.

12. Select the flash devices and the sectors you want to add the erase action to.
13. Click **Add Erase Action**.
14. Click **Done**.

    You return to the <target> **Flash Programmer Task** editor window, and the action appears in the **Flash Programmer Actions** group.



HCS08/RS08/ColdFire V1 Flash Programmer Task

Flash Devices

Device Name	Base Address
MC9S08QG8_FLASH (8kx16x1)	0x0000E000

Target RAM

Address: 0x 80

Size: 0x 200

☐ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	MC9S08QG8_FLASH using chip erase.

Figure 17-24. <target> Flash Programmer Task Editor Window Settings for DEMO9S08QG8

15. Click the **Close** icon to save the settings and close the <target> **Flash Programmer Task** editor window.

    You can access the newly-made `Erase Flash DEMO9S08QG8` task from the **Target Tasks** view.

17.2.3.2 Execute HCS08 Erase Task

To erase the Flash memory on the MC9S08QG8, you use the Erase Flash DEMO9S08QG8 task that you made in the previous topic.

To execute the task and erase the memory:

1. Go the **Target Tasks** view and right-click on the `Erase Flash DEMO9S08QG8` task.
2. Select **Execute** . Alternatively, select the `Erase Flash DEMO9S08QG8` task and click the **Execute** icon on the **Target Tasks** view toolbar.

3. In the **Console** view, status messages appear as the IDE connects to the board and erases the memory.

Congratulations! You have erased the on-chip Flash memory in the DEMO9S08QG8 board's microcontroller.

17.2.4 Tutorial D: Create Erase Flash Memory Task for ColdFire

The goal of this tutorial is to demonstrate how to use the <target> **Flash Programmer Task** editor window to create a task that erases specific topics of Flash memory in a ColdFire MCF5213.

- Set Up ColdFire Erase Task
- Execute ColdFire Erase Task

17.2.4.1 Set Up ColdFire Erase Task

> **NOTE**
> This tutorial assumes that you have already built a ColdFire project with the name ColdFire_test. It also assumes that you have previously created other tasks, so that the **Target Tasks** view is visible. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

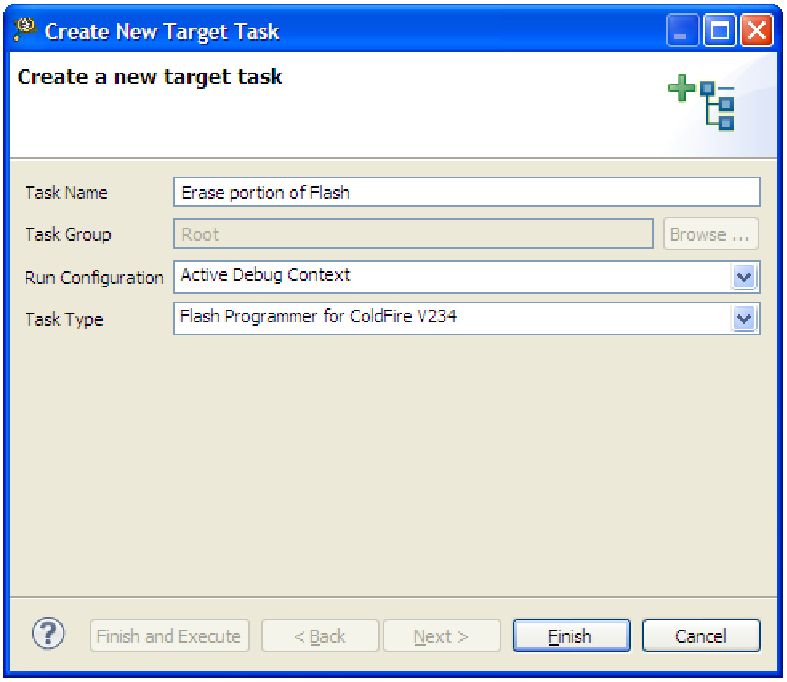
After you have connected an MCF5213-based evaluation board to the workstation with a USB cable:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **New Task**.

   The **Create New Target Task** wizard appears.

3. In the **Task Name** text box, enter the name of the target task. For example, enter Erase portion of Flash.
4. From the **Run Configuration** drop-down list, select a run configuration.
5. From the **Task Type** drop-down list, select **Flash Programmer for ColdFire V234** from the drop-down listbox.

   The **Create New Target Task** wizard should appear.



Create New Target Task

Create a new target task

Task Name: Erase portion of Flash

Task Group: Root

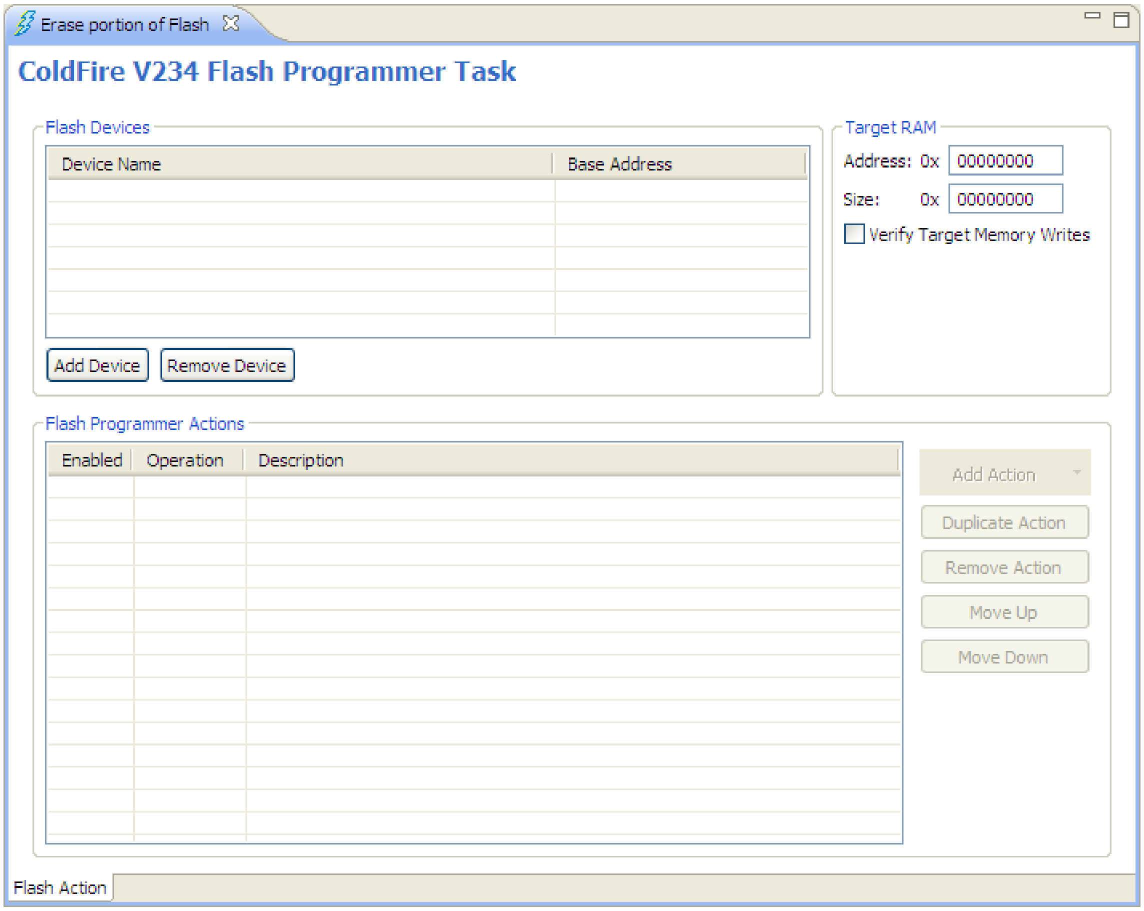
Run Configuration: Active Debug Context

Task Type: Flash Programmer for ColdFire V234

Figure 17-25. Task Settings for Erasing Flash on MCF5213

6. Click **Finish** to create the task.

   The <target> **Flash Programmer Task** editor window appears.



Address: 0x 00000000

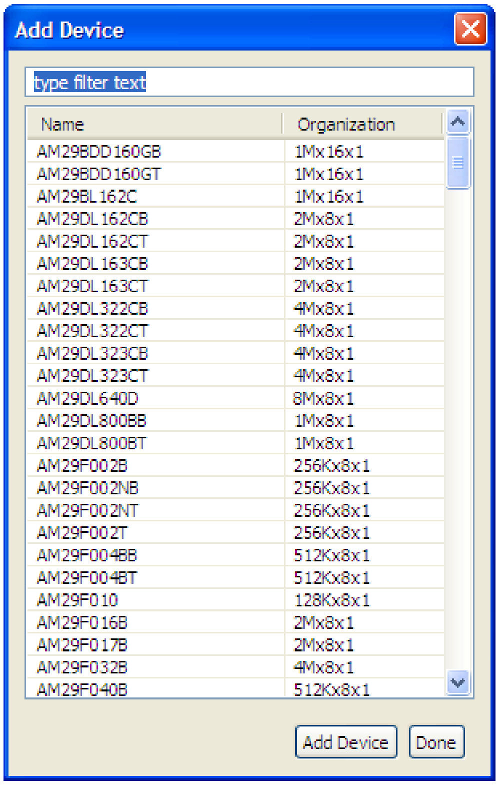
Size: 0x 00000000

□ Verify Target Memory Writes

Figure 17-26. <target> Flash Programmer Task Editor Window

7. Click **Add Device**.

   The **Add Device** dialog box appears.



Name	Organization
AM29BDD160GB	1Mx16x1
AM29BDD160GT	1Mx16x1
AM29BL162C	1Mx16x1
AM29DL162CB	2Mx8x1
AM29DL162CT	2Mx8x1
AM29DL163CB	2Mx8x1
AM29DL163CT	2Mx8x1
AM29DL322CB	4Mx8x1
AM29DL322CT	4Mx8x1
AM29DL323CB	4Mx8x1
AM29DL323CT	4Mx8x1
AM29DL640D	8Mx8x1
AM29DL800BB	1Mx8x1
AM29DL800BT	1Mx8x1
AM29F002B	256Kx8x1
AM29F002NB	256Kx8x1
AM29F002NT	256Kx8x1
AM29F002T	256Kx8x1
AM29F004BB	512Kx8x1
AM29F004BT	512Kx8x1
AM29F010	128Kx8x1
AM29F016B	2Mx8x1
AM29F017B	2Mx8x1
AM29F032B	4Mx8x1
AM29F040B	512Kx8x1

Figure 17-27. Add Device for ColdFire Derivative

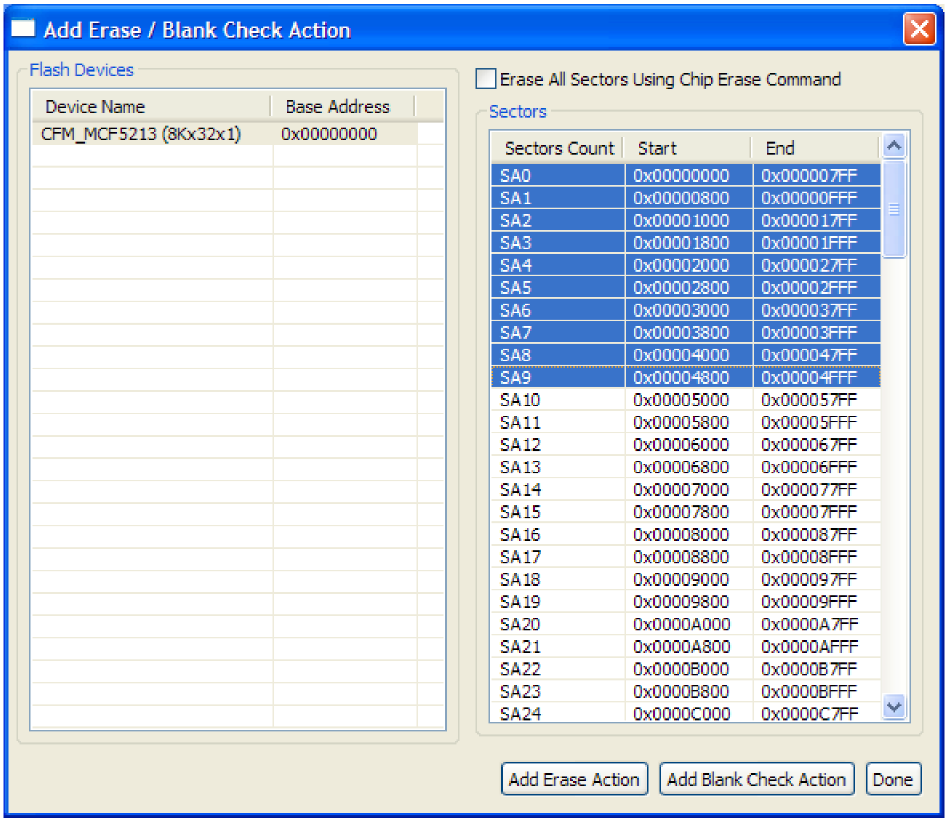
8. Scroll through the list of microcontroller derivatives and select CFM_MCF5213.
9. Click **Add Device**.
10. Click **Done**.

    You return to the <target> **Flash Programmer Task** editor window, with the selected device appearing in the **Flash Devices** group.

11. Select the **Erase / Blank Check** action from the **Add Action** drop-down list.

    The **Add Erase / Blank Check Action** dialog box appears. It displays the flash devices and their addresses.

12. In the **Sectors** group, click on the desired start address of the flash memory, then shift-click on the end address to select all of the sectors of Flash memory you want erased.



Device Name	Base Address
CFM_MCF5213 (8Kx32x1)	0x00000000

□ Erase All Sectors Using Chip Erase Command

Sectors Count	Start	End
SA0	0x00000000	0x000007FF
SA1	0x00000800	0x00000FFF
SA2	0x00001000	0x000017FF
SA3	0x00001800	0x00001FFF
SA4	0x00002000	0x000027FF
SA5	0x00002800	0x00002FFF
SA6	0x00003000	0x000037FF
SA7	0x00003800	0x00003FFF
SA8	0x00004000	0x000047FF
SA9	0x00004800	0x00004FFF
SA10	0x00005000	0x000057FF
SA11	0x00005800	0x00005FFF
SA12	0x00006000	0x000067FF
SA13	0x00006800	0x00006FFF
SA14	0x00007000	0x000077FF
SA15	0x00007800	0x00007FFF
SA16	0x00008000	0x000087FF
SA17	0x00008800	0x00008FFF
SA18	0x00009000	0x000097FF
SA19	0x00009800	0x00009FFF
SA20	0x0000A000	0x0000A7FF
SA21	0x0000A800	0x0000AFFF
SA22	0x0000B000	0x0000B7FF
SA23	0x0000B800	0x0000BFFF
SA24	0x0000C000	0x0000C7FF

Figure 17-28. Add Erase / Blank Action Dialog Box for ColdFire

13. Click **Add Erase Action**.
14. Click **Done**.

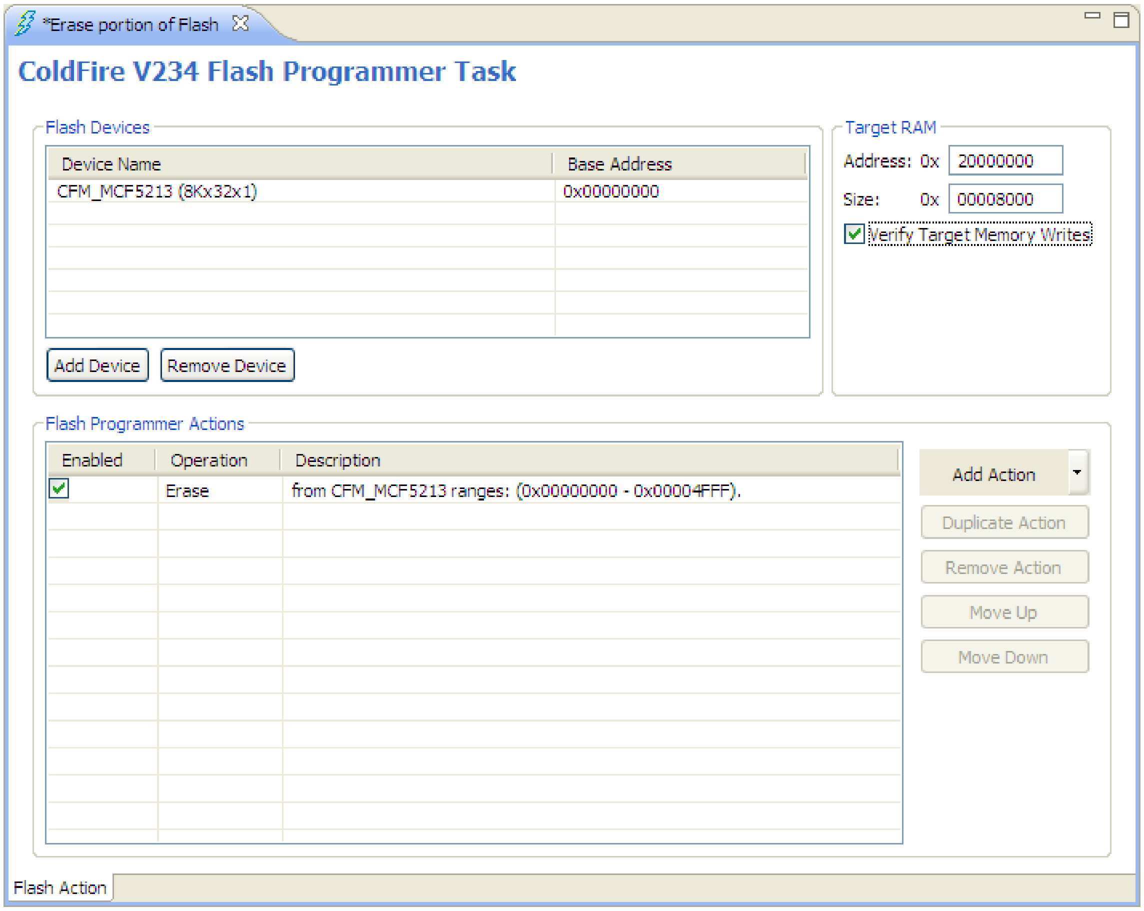
NOTE

To erase of all Flash memory at the same time, check **Erase All Sectors Using Chip Erase Command**.

You return to the <target> **Flash Programmer Target** editor window, and the action appears in the **Flash Programmer Actions** group.

15. To allocate a buffer of RAM to hold the erasure algorithm.
    a. Enter the start address of the RAM buffer in **Address** text box of the **Target RAM** group.
    b. In the **Size** option, enter the amount of RAM that makes up the buffer.
    c. Check the **Verify Target Memory Writes** checkbox. For this example, the start address of the buffer was 0X20000000, and its size was 0X00008000.

    The **Target RAM** group displays the start address of the RAM memory buffer, and its size.



ColdFire V234 Flash Programmer Task

Device Name	Base Address
CFM_MCF5213 (8Kx32x1)	0x00000000

Address: 0x 20000000

Size: 0x 00008000

☑ Verify Target Memory Writes

Enabled	Operation	Description
☑	Erase	from CFM_MCF5213 ranges: (0x00000000 - 0x00004FFF).

Figure 17-29. Flash Programmer Task Editor Window with ColdFire Erase Settings

16. Click the **Close** icon to save the settings and close the <target> **Flash Programmer Task** editor window.

    You can access the newly-made `Erase portion of Flash` task from the **Target Tasks** view.

17.2.4.2 Execute ColdFire Erase Task

To erase the Flash memory on the MCF5213, you use the task that you made in the previous topic.

To execute the task:

1. Go the **Target Tasks** view.
2. Right-click on the **Erase portion of Flash** task.
3. Select **Execute**. Alternatively, click the **Execute** icon on the **Target Tasks** view toolbar.

In the **Console** view, status messages appear as the IDE connects to the board and erases the flash memory.

You have erased the selected on-chip Flash memory sectors in the MCF5123 microcontroller.

17.2.5 Tutorial E: Create Download Program Task for ColdFire

The goal of this tutorial is to demonstrate how to create a task that downloads a program into the ColdFire microcontroller's flash memory before being debugged.

- Set Up Download Task
- Execute ColdFire Program Task

17.2.5.1 Set Up Download Task

NOTE

This tutorial assumes that you have already built a ColdFire project with the name `ColdFire_test`. It also assumes that you have created the `Erase portion of Flash` task from the previous tutorial.

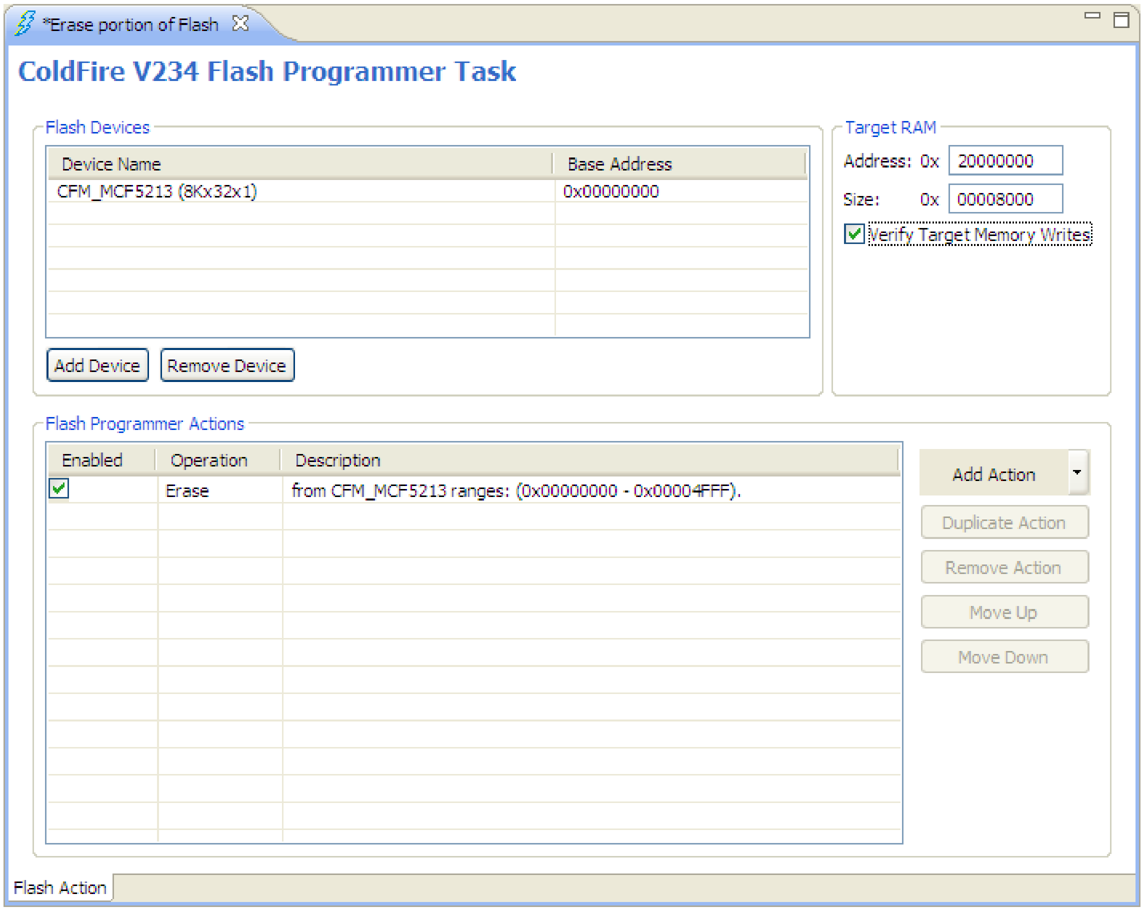
NOTE

To avoid the microcontroller getting caught in an indeterminant state, it is important that its flash memory be erased before attempting to program it. The erase memory action of this task will erase the microcontroller's flash memory before the second action, created in this topic, downloads a program into it. Do not attempt to program flash memory without erasing it first.

After you have connected an MCF5213-based evaluation board to the workstation with a USB cable:

1. Go to the **Target Tasks** view in either the **C/C++ or Debug** perspective.
2. Right-click on this view and double-click on the task Erase portion of Flash that you made in the previous topic.

   The **ColdFire V234 Flash Programmer Task** editor window appears.



*Erase portion of Flash

ColdFire V234 Flash Programmer Task

Flash Devices

Device Name	Base Address
CFM_MCF5213 (8Kx32x1)	0x00000000

Target RAM

Address: 0x 20000000

Size: 0x 00008000

☑ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	from CFM_MCF5213 ranges: (0x00000000 - 0x00004FFF).

Figure 17-30. Task Settings for Erasing Memory on MCF5213

3. Allocate a buffer of RAM that holds the programming algorithm, along with any program code as it is written into flash.
   a. Enter the start address of the RAM buffer in **Address** text box of the **Target RAM** group.
   b. In the **Size** text box, enter the amount of RAM that makes up the buffer.
   c. Check the **Verify Target Memory Writes** option. For this example, the start address of the buffer was `0x20000000`, and its size was `0x00008000`.

   NOTE

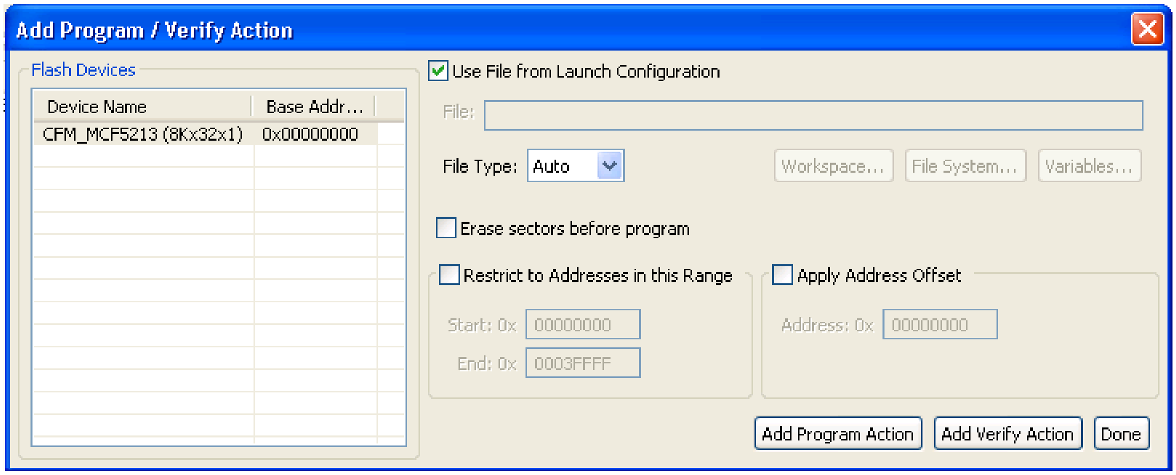
   Since this configuration was taken care of when the Erase action was set up, you do not have to enter anything for this step. However, it is described here for the sake of completeness.

   The **Target RAM** group displays the start address of the RAM memory buffer, and its size.

4. Select the **Program / Verify** action from the **Add Action** drop-down list.

The **Add Program / Verify Action** dialog box appears, with the ColdFire device selected.

5. Check **Use File from Launch Configuration** to use the default .elf file made by the project.



Add Program / Verify Action

Flash Devices

Device Name	Base Addr...
CFM_MCF5213 (8Kx32x1)	0x00000000

☑ Use File from Launch Configuration

File:

File Type: Auto

☐ Erase sectors before program

☐ Restrict to Addresses in this Range

Start: 0x 00000000

End: 0x 0003FFFF

☐ Apply Address Offset

Address: 0x 00000000

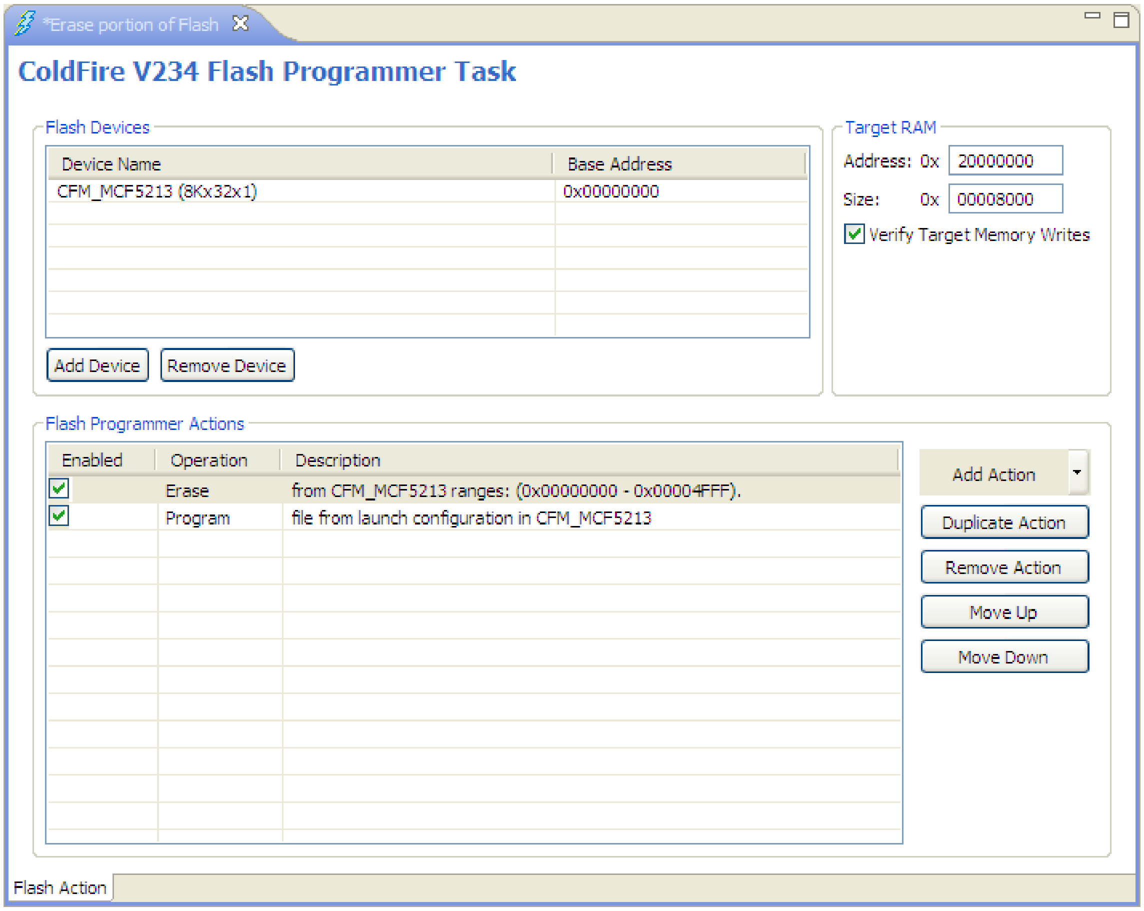
Figure 17-31. Adding File to Add Program / Verify Action Dialog Box

6. Click **Add Program Action**.
7. Click **Done**.

You return to the <target> **Flash Programmer Target** editor window, with the program action appearing in the **Flash Programmer Actions** group.

NOTE

If you want to download a file other than the launch configuration's default file, click on the **Workspace**, **File System**, or **Variables** button and navigate to the file.



ColdFire V234 Flash Programmer Task

Flash Devices

Device Name	Base Address
CFM_MCF5213 (8Kx32x1)	0x00000000

Target RAM

Address: 0x 20000000

Size: 0x 00008000

☑ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Erase	from CFM_MCF5213 ranges: (0x00000000 - 0x00004FFF).
☑	Program	file from launch configuration in CFM_MCF5213

Figure 17-32. Settings for Downloading Program to ColdFire Microcontroller

8. Click the **Close** icon to save the settings and close the <target> **Flash Programmer Task** editor window.

   The revised `Erase portion of Flash` task is available from the **Target Tasks** view.

17.2.5.2 Execute ColdFire Program Task

To execute this task:

1. Go the **Target Tasks** view and right-click on the `MCF5213_INTFLASH` task.
2. Select **Execute** . Alternatively, select the `MCF5213_INTFLASH` task and click the **Execute** icon on the **Target Tasks** view toolbar.
3. In the **Console** view, status messages appear as the IDE connects to the board and erases the memory.

Congratulations! You have erased the on-chip Flash memory in the DEMO9S08QG8 board's microcontroller.

17.2.6 Tutorial F: Import and Execute Power Architecture Flash Task

The goal of this tutorial is to select and import a pre-defined Power Architecture flash task.

- Import Power Architecture Program Flash Task
- Execute Predefined Task

17.2.6.1 Import Power Architecture Program Flash Task

To import a pre-defined Power Architecture flash task:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **Import** . Alternatively, click the **Import** icon on the **Target Tasks** view toolbar.

   The **Open** dialog box appears.

3. Navigate to `<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/E200` and select the XML file for the board's microcontroller.

   NOTE

   One processor has multiple flash areas that are treated as different devices. Therefore, you should be very sure of the needs when selecting a task.

   There are predefined target tasks for each flash module. However, depending on processor type some of them might not be available.

   - Internal Flash - The Internal Flash target task is for Code Flash and Data Flash if they exist or for the user area of Internal Flash. Comprises everything - the area for application (sometimes has code + data), shadow flash and tester areas.
     - Code Flash - Program's code should be programmed here.
     - Data flash - Flash special design to hold data.
       - Shadow Flash - Flash that contains system configuration options besides the user area. Erasing it can also change the system state after reset. Some of these registers are for device's security and you cannot write anything there. Additionally, the erase sequence is different. After you

erase the device, the security information is programmed in the flash and thus the blank check fails.

- Tester Areas - Special flash that cannot be erased and also have registers that can change processor's behavior like which flash sectors are protected. Changing them cannot be reverted, which means that the Flash is written once.

NOTE

Some processors can interpret two sets of instructions: BOOKE and VLE. If the processor supports both of them target tasks are provided for each instruction set. The user should choose the right one.

- Click **Open**.

  The selected task appears in the **Target Tasks** view.

- Double-click on the task to examine its contents.

  The <target> **Flash Programmer Task** editor window appears, and displays the memory settings and actions for the task. Notice the actions to erase, program, and/or verify the contents of flash memory in the **Flash Programmer Actions** group.

17.2.6.2 Execute Predefined Task

To execute the task:

1. Right-click on the imported task.
2. Select **Execute** . Alternatively, select the imported task and click the **Execute** icon on the **Target Tasks** view toolbar.

   CodeWarrior establishes contact with the board, erases the microcontroller's flash memory, downloads the code, and verifies that the contents of flash match those of the file.

Congratulations! You have selected and used a target task that erased and programmed the flash memory on the pre-defined Power Architecture task.

17.2.7 Tutorial G: Switching Between Lock-Step and Decoupled Parallel Modes

The goal of this tutorial is to demonstrate the steps to switch between the **Lock-Step Mode (LSM)** and **Decoupled Parallel Mode (DPM)** .

- Import DPM Target Task
- Execute Predefined DPM Task
- Hardware Reset
- Import LSM Target Task
- Execute Predefined LSM Task

17.2.7.1 Import DPM Target Task

NOTE

> This procedure assumes that the **Target Tasks** view is visible in the perspective. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

After you have launched CodeWarrior and the machine has a USB NEXUS Multilink connected to a Leopard (for example, MPC5643L) DPM board:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **Import** . Alternatively, click the **Import** icon on the **Target Tasks** view toolbar.

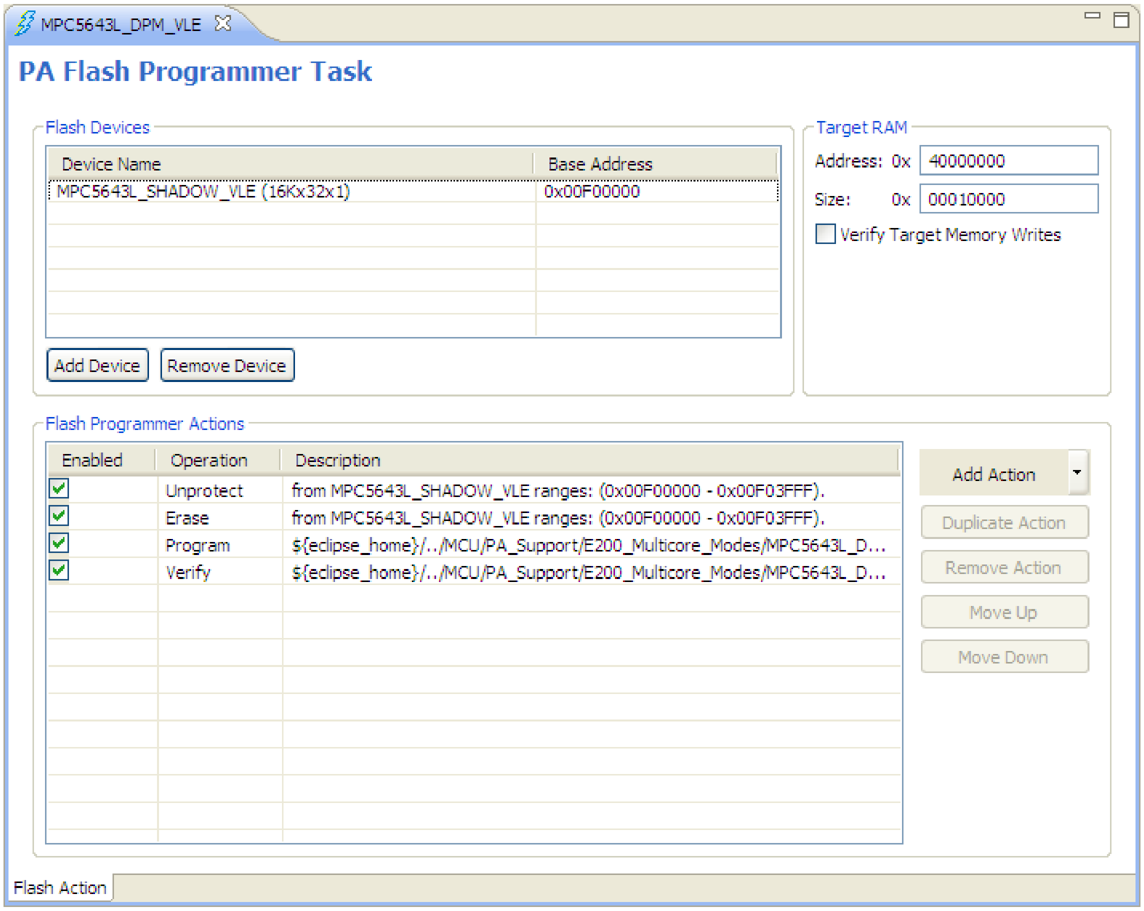
   The **Open** dialog box appears.

3. Navigate to `<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/E200` and select the XML file for the board's microcontroller. For this tutorial, select `MPC5643L_DPM_VLE.xml`.
4. Click **Open**.

   The `MPC5643L_DPM_VLE` task appears in the **Target Tasks** view.

5. Double-click on the `MPC5643L_DPM_VLE` to examine its contents.

   The <target> **Flash Programmer Task** editor window appears, and displays the memory settings and actions for the task. Notice the actions to erase, program, and verify the contents of flash memory in the **Flash Programmer Actions** group. These actions execute in the order as they are displayed in the table, from top to bottom.



PA Flash Programmer Task

Device Name	Base Address
MPC5643L_SHADOW_VLE (16Kx32x1)	0x00F00000

Address: 0x 40000000

Size: 0x 00010000

□ Verify Target Memory Writes

Enabled	Operation	Description
☑	Unprotect	from MPC5643L_SHADOW_VLE ranges: (0x00F00000 - 0x00F03FFF).
☑	Erase	from MPC5643L_SHADOW_VLE ranges: (0x00F00000 - 0x00F03FFF).
☑	Program	${eclipse_home}/../MCU/PA_Support/E200_Multicore_Modes/MPC5643L_D...
☑	Verify	${eclipse_home}/../MCU/PA_Support/E200_Multicore_Modes/MPC5643L_D...

Figure 17-33. Settings for MPC5643L_DPM_VLE Target Task

17.2.7.2 Execute Predefined DPM Task

To execute the task:

1. Right-click on the imported task.
2. Select **Execute** . Alternatively, select the imported task and click the **Execute** icon on the **Target Tasks** view toolbar.

   CodeWarrior establishes contact with the board, erases the microcontroller's flash memory, downloads the code, and verifies that the contents of flash match those of the file.

17.2.7.3 Hardware Reset

To reset the hardware:

1. Unplug the power cord.
2. Plug the power cord again.

   or

   Simply press the reset button on the hardware or target board.

Congratulations! You have selected and used a target task that erased and programmed the flash memory on the pre-defined Power Architecture task.

17.2.7.4 Import LSM Target Task

> **NOTE**
>
> This procedure assumes that the **Target Tasks** view is visible in the perspective. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

After you have launched CodeWarrior and the machine has a USB NEXUS Multilink connected to a Leopard (for example, MPC5643L) LSM board:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click on this view and select **Import** . Alternatively, click the **Import** icon on the **Target Tasks** view toolbar.

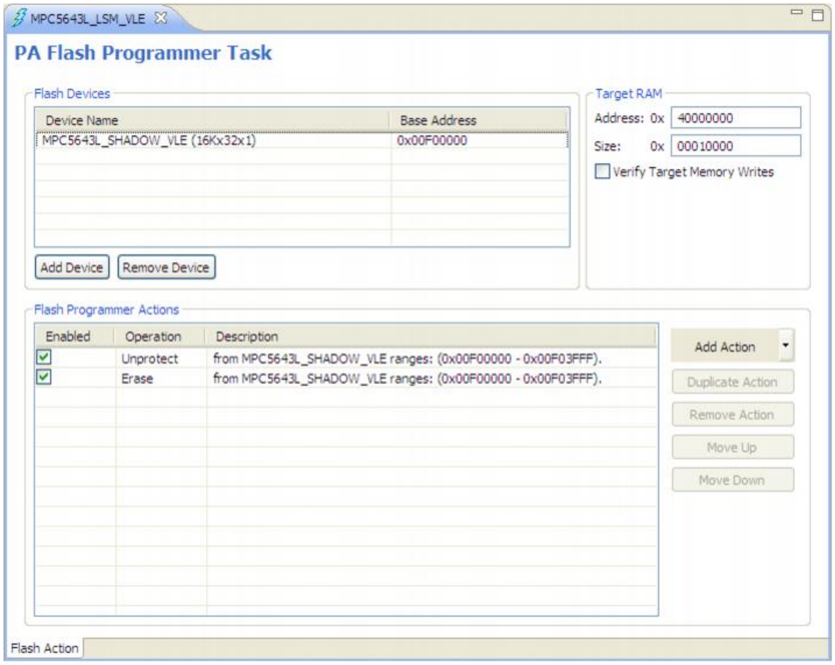
   The **Open** dialog box appears.

3. Navigate to `<CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/E200` and select the XML file for the board's microcontroller. For this tutorial, select `MPC5643L_LSM_VLE.xml`.
4. Click **Open**.

   The `MPC5643L_LSM_VLE` task appears in the **Target Tasks** view.

5. Double-click on the `MPC5643L_LSM_VLE` to examine its contents.

   The <target> **Flash Programmer Task** editor window appears, and displays the memory settings and actions for the task. Notice the actions to erase the contents of flash memory in the **Flash Programmer Actions** group. These actions execute in the order as they are displayed in the table, from top to bottom.



PA Flash Programmer Task

Flash Devices

Device Name	Base Address
MPC5643L_SHADOW_VLE (16Kx32x1)	0x00F00000

Target RAM

Address: 0x 40000000

Size: 0x 00010000

☐ Verify Target Memory Writes

Flash Programmer Actions

Enabled	Operation	Description
☑	Unprotect	from MPC5643L_SHADOW_VLE ranges: (0x00F00000 - 0x00F03FFF).
☑	Erase	from MPC5643L_SHADOW_VLE ranges: (0x00F00000 - 0x00F03FFF).

Figure 17-34. Settings for MPC5643L_LSM_VLE Target Task

17.2.7.5 Execute Predefined LSM Task

To execute the task:

1. Right-click on the imported task.
2. Select **Execute** . Alternatively, select the imported task and click the **Execute** icon on the **Target Tasks** view toolbar.

   CodeWarrior establishes contact with the board, erases the microcontroller's flash memory.

Congratulations! You have switched between the Lock-Step and Decoupled Parallel Modes.

17.2.8 Tutorial H: Create and Execute Diagnostics Action Task

The goal of this tutorial is to demonstrate the steps to create and execute the diagnostics action using the flash programmer.

- Set Up Diagnostics Action Task
- Execute Diagnostics Action Task

17.2.8.1 Set Up Diagnostics Action Task

NOTE

This procedure assumes that you have imported a pre-defined Flash Programmer task in the **Target Tasks** view. It also assumes that you have previously created other tasks, so that the **Target Tasks** view is visible. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the **Target Tasks** view.

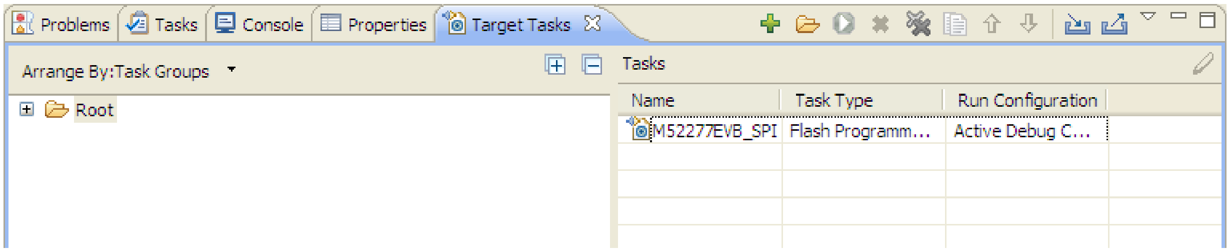
After you have launched CodeWarrior and connected the `M52277EVB_SPI` board to the host system using a USB cable:

1. Go to the **Target Tasks** view in either the **C/C++** or **Debug** perspective.
2. Right-click in the **Target Tasks** view and select **Import** . Alternatively, click the **Import** icon on the **Target Tasks** view toolbar.

   The **Open** dialog box appears.

3. Navigate to the pre-defined tasks folder at `<CW MCU install>\MCU\bin\plugins\support\TargetTask\Flash_Programmer\` and select the desired .xml file for your hardware target. For example, select `M52277EVB_SPI.xml`.
4. Click **Open**.

   The selected task appears in the **Target Tasks** view.



Name	Task Type	Run Configuration
M52277EVB_SPI	Flash Programm...	Active Debug C...

Figure 17-35. Pre-defined Task in Target Tasks View

NOTE

The predefined erase/program tasks are not mandatory for diagnostics.

5. Right-click on the task's name and select **Execute**.

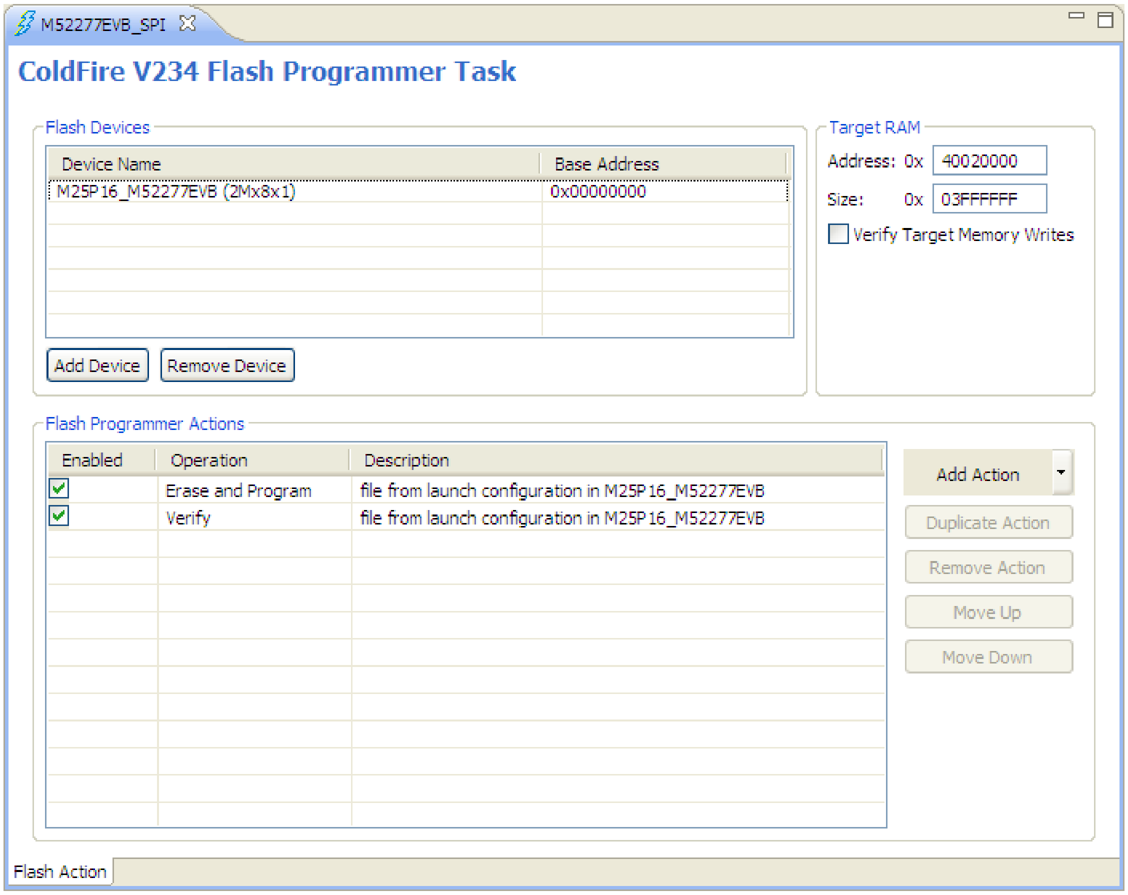
The task's Flash Programmer actions execute in sequence. First, they erase the hardware target's flash memory. Next, they check whether the flash is correctly erased. If true, they program the file's code into the flash and check if it programmed without errors.

NOTE

When a predefined flash programmer task is imported, its **Run Configuration** is set as Active Debug Context. If the task is imported and there is no active debug session, then the **Execute** icon will be disabled as in. Associate the selected target task to a different Run Configuration to enable the **Execute** icon.

6. Double-click on the task name, to examine the task stored Flash Programmer actions.

   The <target> **Flash Programmer Task** editor window appears.



ColdFire V234 Flash Programmer Task

Device Name	Base Address
M25P16_M52277EVB (2Mx8x1)	0x00000000

Address: 0x 40020000

Size: 0x 03FFFFFF

☐ Verify Target Memory Writes

Enabled	Operation	Description
☑	Erase and Program	file from launch configuration in M25P16_M52277EVB
☑	Verify	file from launch configuration in M25P16_M52277EVB

Figure 17-36. <target> Flash Programmer Task Editor Window

7. Select the **Diagnostics** action from the **Add Action** drop-down list.

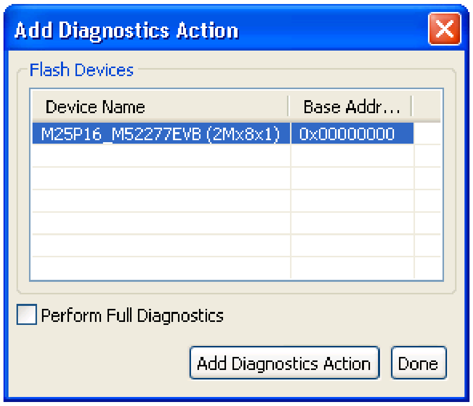
   The **Add Diagnostics Action** dialog box appears. It displays the flash devices and the base addresses.

NOTE

The full diagnostics does the same thing as diagnostics but also prints the blank status for sectors. It can take significantly longer to complete.

NOTE

If more than one flash is available in **Flash Devices** table, the **Add Diagnostics Action** table lets you select the flash where you want to run the diagnostics.



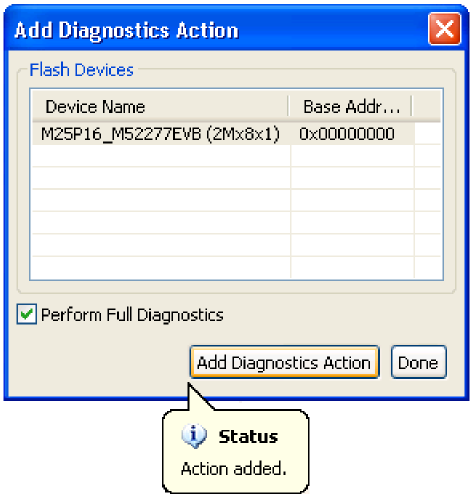
Device Name	Base Addr...
M25P16_M52277EVB (2Mx8x1)	0x00000000

□ Perform Full Diagnostics

Figure 17-37. Add Diagnostics Action Dialog Box

8. Check the **Perform Full Diagnostics** checkbox if you want to perform complete diagnostics on the selected flash device.
9. Select the **Diagnostics** action from the **Add Action** drop-down list.

   You get a popup with a status that the device is added.



Device Name	Base Addr...
M25P16_M52277EVB (2Mx8x1)	0x00000000

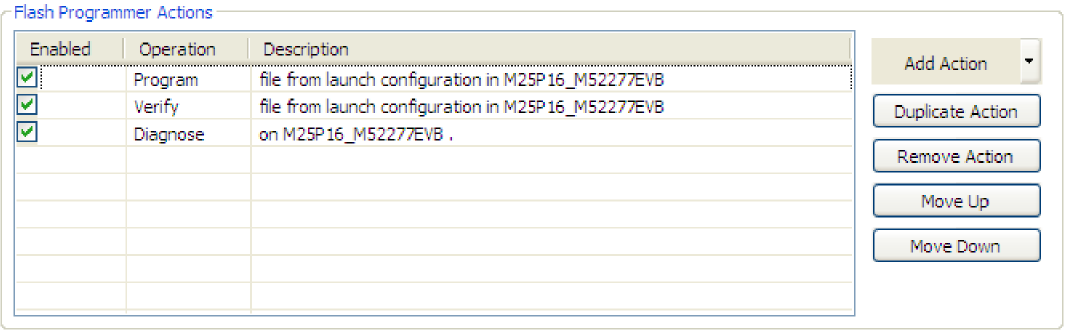
☑ Perform Full Diagnostics

Status
Action added.

Figure 17-38. Add Diagnostics Actions Dialog Box - Popup with Status

10. Click **Done**.

The **Add Diagnostics Action** dialog box closes. The action appears in the **Flash Programmer Actions** table of the <target> **Flash Programmer Task** editor window.



Enabled	Operation	Description
☑	Program	file from launch configuration in M25P16_M52277EVB
☑	Verify	file from launch configuration in M25P16_M52277EVB
☑	Diagnose	on M25P16_M52277EVB .

Figure 17-39. Flash Programmer Actions Table

17.2.8.2 Execute Diagnostics Action Task

To execute the task:

1. Right-click on the `M52277EVB_SPI` task.
2. Select **Execute**. Alternatively, select the `M52277EVB_SPI` task and click the **Execute** icon on the **Target Tasks** view toolbar.

   CodeWarrior establishes contact with the `M52277EVB_SPI` board, erases the microcontroller's flash memory, downloads the code, verifies, and diagnoses the contents on `M52277EVB_SPI`.

You have created and executed a diagnostic action task on the `M52277EVB_SPI` microcontroller.

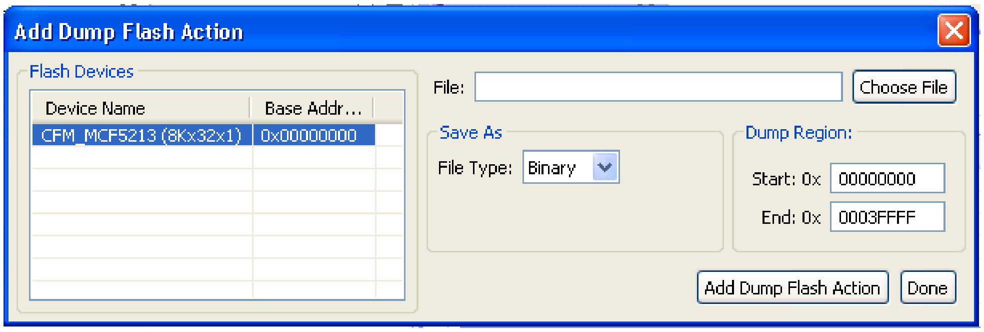
17.2.9 Tutorial I: Dump Entire Flash

The goal of this tutorial is to demonstrate how to dump selected sectors of a flash device or the entire flash device.

To add a dump flash action:

1. Select the **Dump Flash** action from the **Add Action** drop-down list.

   The **AddDump Flash Action** dialog box appears.



Device Name	Base Addr...
CFM_MCF5213 (8Kx32x1)	0x00000000

File:

File Type: Binary

Start: 0x 00000000

End: 0x 0003FFFF

Figure 17-40. Add Dump Flash Action Dialog Box

2. Specify the file name in the **File** text box. The flash is dumped in this selected file.
3. Select the file type from the **File Type** drop-down list. You can select any one of the following file types:
   - Srec - Saves files in Motorola S-record format.
   - Binary - Saves files in binary file format.
4. Specify the memory range for which you want to add dump flash action.
   - Specify the start address of the range in the **Start** text box.
   - Specify the end address of the range in the **End** text box.
5. Click **Add Dump Flash Action**.
6. Click **Done**.

   The **Add Dump Flash Action** dialog box closes and the added dump flash action appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.

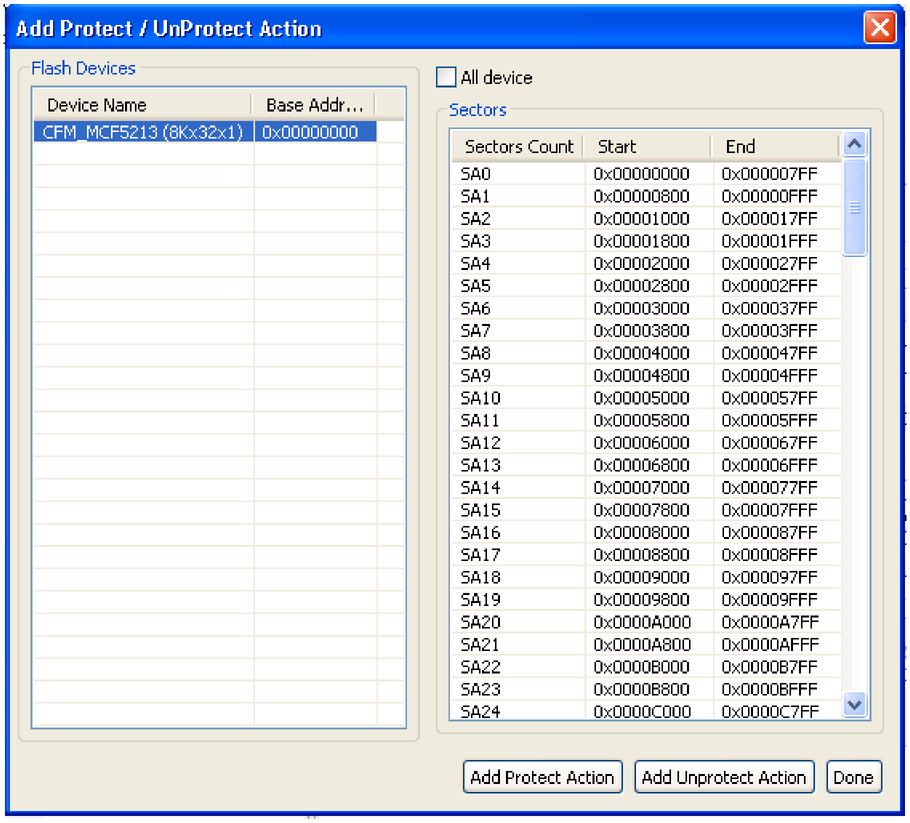
17.2.10 Tutorial J: Change Protection of Sector

The goal of this tutorial is to demonstrate how to protect / unprotect actions enable you to change the protection of a sector in the flash device.

To add a protect / unprotect action:

1. Select the **Protect/Unprotect** action from the **Add Action** drop-down list.

   The **Add Protect/Unprotect Action** dialog box appears.



Device Name	Base Addr...
CFM_MCF5213 (8Kx32x1)	0x00000000

☐ All device

Sectors Count	Start	End
SA0	0x00000000	0x000007FF
SA1	0x00000800	0x00000FFF
SA2	0x00001000	0x000017FF
SA3	0x00001800	0x00001FFF
SA4	0x00002000	0x000027FF
SA5	0x00002800	0x00002FFF
SA6	0x00003000	0x000037FF
SA7	0x00003800	0x00003FFF
SA8	0x00004000	0x000047FF
SA9	0x00004800	0x00004FFF
SA10	0x00005000	0x000057FF
SA11	0x00005800	0x00005FFF
SA12	0x00006000	0x000067FF
SA13	0x00006800	0x00006FFF
SA14	0x00007000	0x000077FF
SA15	0x00007800	0x00007FFF
SA16	0x00008000	0x000087FF
SA17	0x00008800	0x00008FFF
SA18	0x00009000	0x000097FF
SA19	0x00009800	0x00009FFF
SA20	0x0000A000	0x0000A7FF
SA21	0x0000A800	0x0000AFFF
SA22	0x0000B000	0x0000B7FF
SA23	0x0000B800	0x0000BFFF
SA24	0x0000C000	0x0000C7FF

Figure 17-41. Add Protect / Unprotect Action Dialog Box

2. Select a sector from the **Sectors** table and click the **Add Protect Action** button to add a protect operation on the selected sector.

NOTE

Press **CTRL** or **SHIFT** keys for selecting multiple sectors from the **Sectors** table.

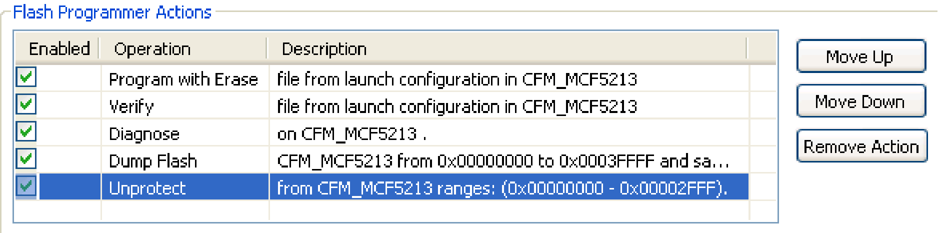
3. Click the **Add Unprotect Action** button to add an unprotect action on the selected sector.

NOTE

Check the **All Device** checkbox to add action on full device.

4. Click **Done**.

The **Add Protect/ Unprotect Action** dialog box closes and the added protect or unprotect actions appear in the **Flash Programmer Actions** table in the **Flash Programmer Task** editor window.



Enabled	Operation	Description
☑	Program with Erase	file from launch configuration in CFM_MCF5213
☑	Verify	file from launch configuration in CFM_MCF5213
☑	Diagnose	on CFM_MCF5213 .
☑	Dump Flash	CFM_MCF5213 from 0x00000000 to 0x0003FFFF and sa...
☑	Unprotect	from CFM_MCF5213 ranges: (0x00000000 - 0x00002FFF).

Figure 17-42. Protect / Unprotect Action

17.2.11 Tutorial K: Fast Access to Target Tasks Editors

The goal of this tutorial is to demonstrate managing and fast accessing of the target tasks' editors used by the Target Task framework.

You can access target tasks by:

- Editing tasks in a project. Refer Editing Tasks in Project.
- Editing tasks imported in a previous session. Refer Editing Tasks Imported in Previous Session.
- Storing task to a file. Refer Storing Task to File.

17.2.11.1 Editing Tasks in Project

NOTE

Before editing ensure that the target task has been added to the project by a wizard or another method and has an extension recognized by the feature (.ttf).

To edit tasks in a project:

1. Double-click on the file in the project.
2. The appropriate task editor appears.
3. The task is imported in the Target Task Framework.

17.2.11.2 Editing Tasks Imported in Previous Session

NOTE

Before editing ensure that a target task has been imported and the previous Eclipse session has been closed.

To edit tasks imported in a previous session:

1. Open the **Target Tasks** view.
2. Double-click to open the desired task.

   The target task editor appears.

3. Make the appropriate changes and close the target task editor.

   NOTE

   Changing the task will also save changes to the file in the project. The save is done to the file in project only if the task has been imported from a project (with double click). Otherwise, it will prompt to save the file.

17.2.11.3 Storing Task to File

NOTE

Before storing the task ensure that the target task has been imported or created and appropriate changes have been made.

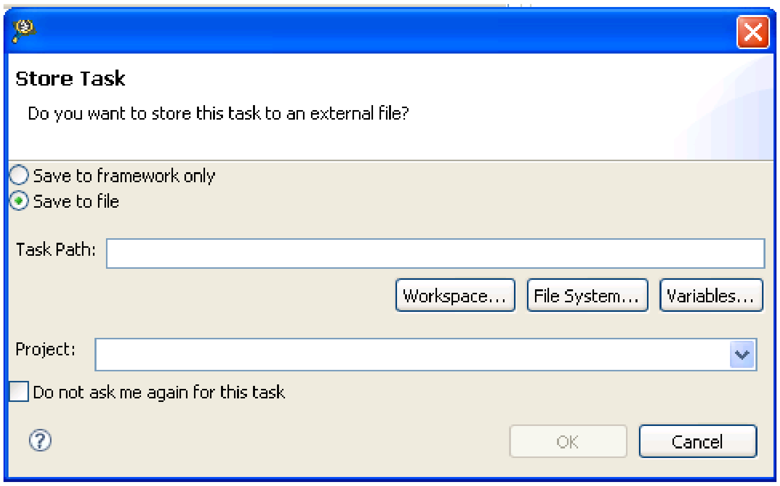
To store task to a file:

1. Open the **Target Tasks** view.
2. Double-click to open the desired task.

   The target task editor appears.

3. Make the appropriate changes and save the target task.

   The **Store Task** dialog box appears.



Store Task

Do you want to store this task to an external file?

○ Save to framework only
● Save to file

Task Path:

Project:

□ Do not ask me again for this task

Figure 17-43. Store Task Dialog Box

4. Select the **Save to File** option.
5. In the **Task Path** text box, specify the path where you want to store the task. You can use the **Workspace**, **File System**, or **Variables** buttons to navigate to the desired location.
6. From the **Project** drop-down list select the project where you want to store you target task.

NOTE

Check the **Do not ask me again for this task** checkbox to save these settings for the current target task.

7. Click **OK**.

The dialog box closes and associates the file to the specified project and saves in target task framework and not necessarily in the project.

NOTE

The above mentioned feature has a preference that will not display the save as dialog and always save in target task framework. The settings are located in **Windows > Preferences > C/C++ >Debug > CodeWarrior Debugger > Show "Save As"** dialog box when saving a new task.

17.2.12 Tutorial L: Programming with Simple Flash

The goal of this tutorial is to demonstrate the use of CodeWarrior Simple Flash Programmer. This feature enables you to perform these basic flash operations:

- Erasing Flash Device
- Programming File

To open the **Simple Flash** dialog box:

1. Click the **Flash Programmer** icon on the IDE toolbar.

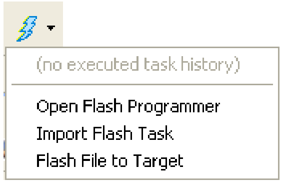
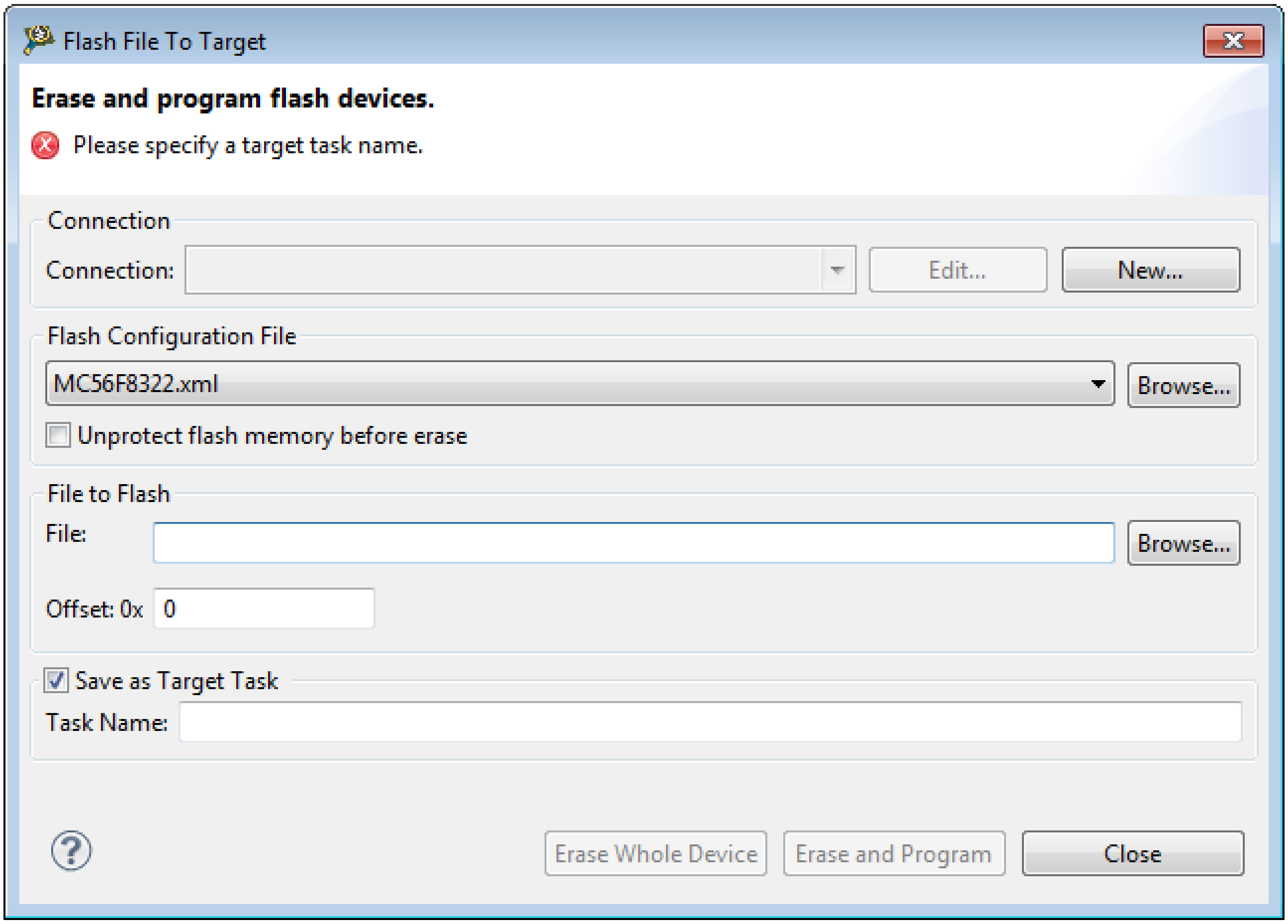


Figure 17-44. Flash Programmer Icon

2. Select **Flash File to Target**.

   The **Simple Flash** dialog box appears.



Erase and program flash devices.

Please specify a target task name.

MC56F8322.xml

□ Unprotect flash memory before erase

Offset: 0x 0

☑ Save as Target Task

Task Name:

Figure 17-45. Simple Flash Dialog Box

- **Remote system Connection drop-down list** - Lists all run configurations defined in Eclipse. If a connection to the target has already been made the control becomes inactive and contains the text Active Debug Configuration.

- **Flash Configuration File pop-up** menu - Lists predefined target tasks for the processor selected in the Launch Configuration and tasks added by user with the **Browse** button. The items in this pop-up menu are updated based on the processor selected in the launch configuration.
    - **Unprotect flash memory before erase** checkbox - Select to unprotect flash memory before erasing the flash device. This feature allows you to unprotect the flash memory from Flash File To Target dialog.
- **File to Flash** group - Allows selecting the file to be programmed on the flash device and the location.
- **File** textbox - Used for specifying the filename. You can use the Workspace, File System, or Variables buttons to select the desired file.
- **Offset:0x** textbox - Used for specifying offset location for a file. If no offset is specified the default value of zero is used. The offset is always added to the start address of the file. If the file does not contain address information then zero is considered as start address.
- **Save as Target Task** - Select to enable Task Name textbox.
    - Task Name textbox - Lets you to save the specified settings as a Flash target task. Use the testbox to specify the name of the target task.
- **Erase Whole Device** button - Erases the flash device. In case you have multiple flash blocks on the device, all blocks are erased. If you want to selectively erase or program blocks, use the Flash programmer feature.
- **Erase and Program** button - Erases the sectors that are occupied with data and then programs the file. If the flash device can not be accessed at sector level then the flash device is completely erased. This feature helps you perform these basic flash operations:

17.2.12.1 Erasing Flash Device

To erase a flash device:

1. Click the **Flash Programmer** icon on the IDE toolbar.

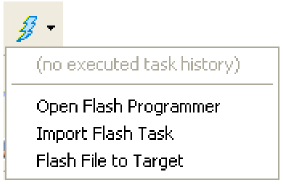


Figure 17-46. Flash Programmer Icon

2. Select **Flash File to Target**.
3. The **Simple Flash** dialog box appears.

4. Select a run configuration from the **Run Configuration** drop-down list.

NOTE

If a connection is already established with the target, this control is disabled.

The **Flash Configuration** drop-down list is updated with the supported configurations for the processor from the launch configuration.

5. Select a flash configuration from the **Flash Configuration** drop-down list.
6. Click the **Erase Device** button.

17.2.12.2 Programming File

1. Click the **Flash Programmer** icon on the IDE toolbar.

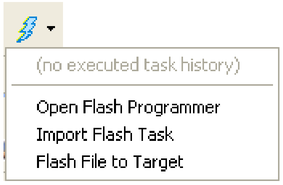


Figure 17-47. Flash Programmer Icon

2. Select **Flash File to Target**.
3. The **Simple Flash** dialog box appears.
4. Select a run configuration from the **Run Configuration** drop-down list.

NOTE

If a connection is already established with the target, this control is disabled.

The **Flash Configuration** drop-down list is updated with the supported configurations for the processor from the launch configuration.

5. Select a flash configuration from the **Flash Configuration** drop-down list.
6. Specify the file name in the **File** text box. You can use the **Workspace**, **File System**, or **Variables** buttons to select the desired file.
7. Specify the offset location in the **Offset** text box.
8. Click the **Program with Erase** button.

17.2.13 Tutorial M: Exporting Target Tasks

The goal of this tutorial is to demonstrate how to export a target task to an external file.

To export a target task:

1. Select the target task in the **Target Task** view.
2. Click the **Export** button from the **Target Task** view toolbar. Alternatively, right-click the target task and select **Export** from the context menu.

   The **Save As** dialog box appears.
3. Specify a file name in the **File name** drop-down list.
4. Click **Save**.

   The exported task is stored in XML format.

17.3 Working with Hardware Diagnostics Window

The **Hardware Diagnostics** window lets you run a series of diagnostic tests that determine if the basic hardware is functional.

These tests include:

- Memory read/write - Makes a read / write access to memory in order to read or write a byte, word (2 bytes), and long word (4 bytes) to / from memory.
- Scope loop - Makes read and write accesses to memory in a loop at target address. The the loop speed settings determine the time between accesses. The loop can only be stopped by cancelling the test.
- Memory tests - Requires you to set the access size and target address from the access settings group and the settings present in the Memory Tests group.

On the Eclipse IDE, the hardware diagnostics feature runs like a target task.

To create a hardware diagnostic target task:

1. Click **Window > Show View > Other**.

   The **Show View** dialog box appears.
2. Expand the **Debug** group and select **Target Tasks**.
3. Click **OK**.
4. Click **Add New Task** from the **Target Tasks** view **t** oolbar to create a new target task.

   The **Create New Target Task** wizard appears.

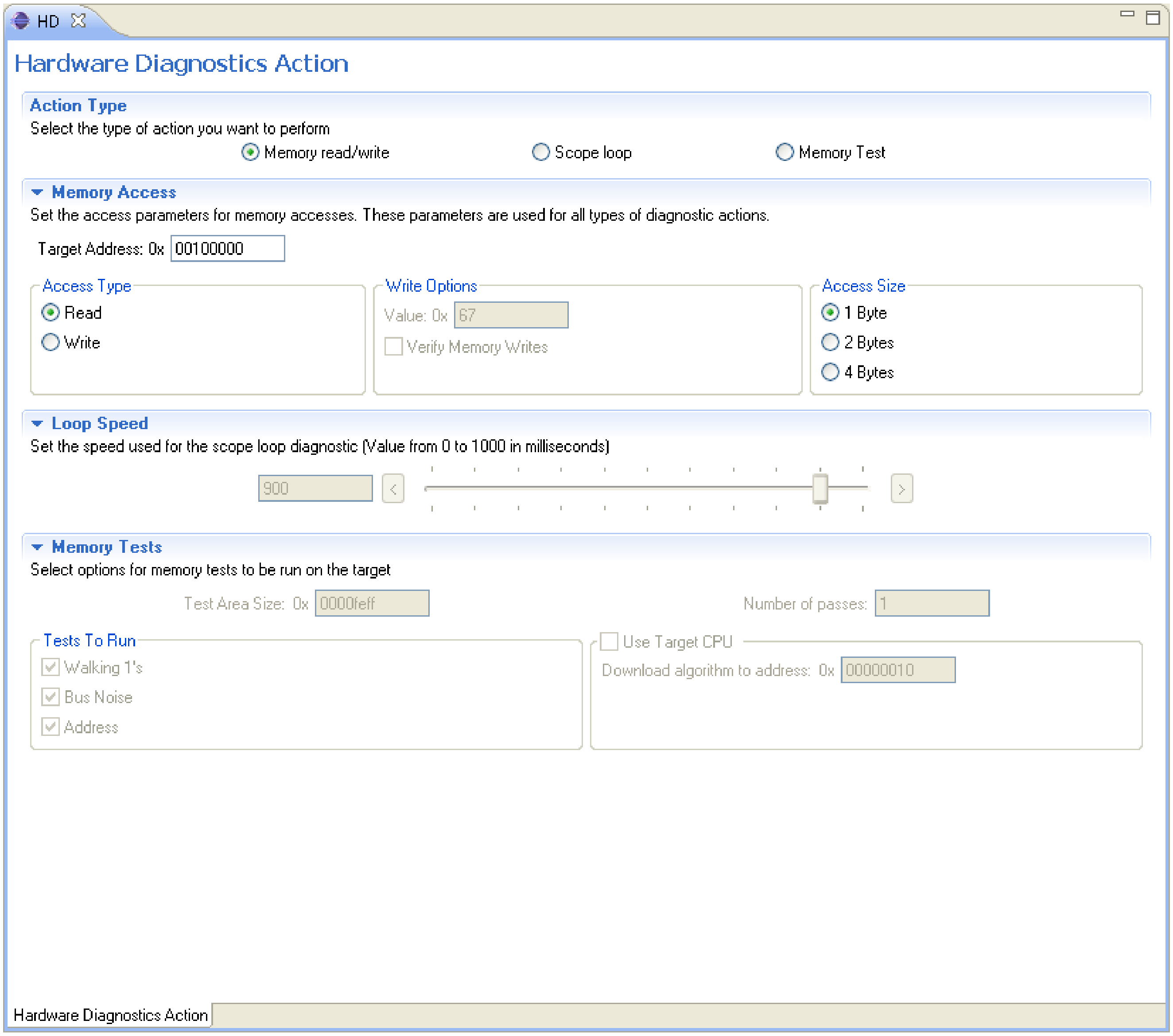
5. In the **Task Name** text box, enter the name of the target task.
6. From the **Run Configuration** drop-down list, select a configuration.

NOTE

Select Active Debug Context from the **Run Configuration** drop-down list, if you want to use hardware diagnostics over an active debugger session, else select any of the specified debug context from the list.

7. From the **Task Type** drop-down list, select **Hardware Diagnostic**.
8. Click **Finish**.

The **Hardware Diagnostics Action** window appears.



HD

Hardware Diagnostics Action

Action Type

Select the type of action you want to perform

● Memory read/write ○ Scope loop ○ Memory Test

Memory Access

Set the access parameters for memory accesses. These parameters are used for all types of diagnostic actions.

Target Address: 0x 00100000

Access Type

● Read

○ Write

Write Options

Value: 0x 67

□ Verify Memory Writes

Access Size

● 1 Byte

○ 2 Bytes

○ 4 Bytes

Loop Speed

Set the speed used for the scope loop diagnostic (Value from 0 to 1000 in milliseconds)

900

Memory Tests

Select options for memory tests to be run on the target

Test Area Size: 0x 0000feff

Number of passes: 1

Tests To Run

☑ Walking 1's

☑ Bus Noise

☑ Address

□ Use Target CPU

Download algorithm to address: 0x 00000010

Hardware Diagnostics Action

Figure 17-48. Hardware Diagnostics Action Window

The **Hardware Diagnostics Action** window includes the following groups:

- **Action Type** - Used to set various action types. The options you select in this group enables the options in the other groups of the window.
- **Memory Access** - Configures diagnostic tests for performing memory reads and writes over the remote connection interface.
- **Loop Speed** - Configures diagnostic tests for performing repeated memory reads and writes over the remote connection interface.
- **Memory Tests** - Configures the memory tests that you can run on the target.

NOTE

> The **Use Target CPU** group appears grayed-out and is not applicable for the HCS08 and RS08 Target Tasks. Refer the *CodeWarrior Common Features Guide* for detailed documentation of the various options available in the **Hardware Diagnostics Action** window.

17.4 Manipulating Target Memory

You can manipulate the target's memory in these ways:

- Import - Read encoded data from a specified file, decode that data into a specific format, and copy the decoded data into a specified memory range. For information, refer Creating Target Task to Import Memory.
- Export - Read data from a specified memory range, encode that data in a specific format, and store the encoded data in an output file. For information, refer Creating Target Task to Export Memory.
- Fill - Fill a specified memory range with a specific data pattern. For information, refer Fill Memory with Data Pattern.

17.4.1 Creating Target Task to Import Memory

Perform these steps to create a target task to import memory:

1. Select **Window** > **Show View** > **Other**.

   The **Show View** dialog box appears.

2. From the **Debug** group, select **Target Tasks**.

The **Target Tasks** view appears.

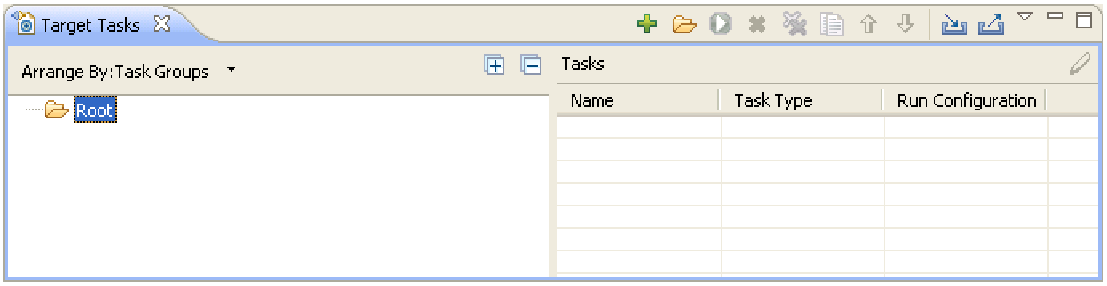


Figure 17-49. Target Tasks View

3. Right-click in the **Target Tasks** view and select **New Task** from the context menu.

   The **Create New Target Task** wizard appears.

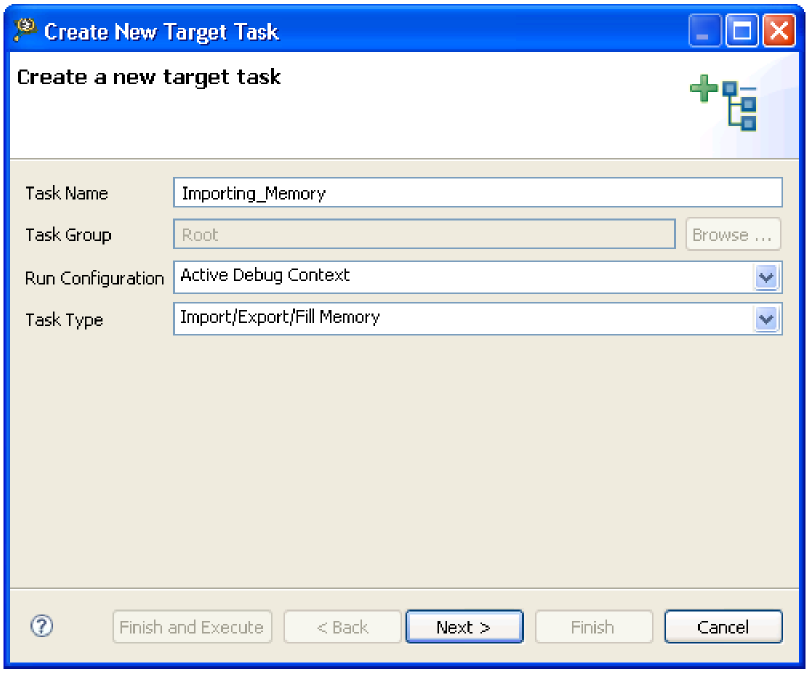
4. In the **Task Name** text box, enter a name for the new task. For example, `Importing_Memory`.

5. Use the **Run Configuration** list box to specify the configuration that the task launches and uses to connect the target. For example, select Active Debug Context.

   NOTE

   If the task does not successfully launch the configuration that you specify, the **Execute** button of the **Target Tasks** view toolbar stays disabled.

6. From the **Task Type** list box, select **Import/Export/Fill Memory**.



Task Name: Importing_Memory
Task Group: Root
Run Configuration: Active Debug Context
Task Type: Import/Export/Fill Memory

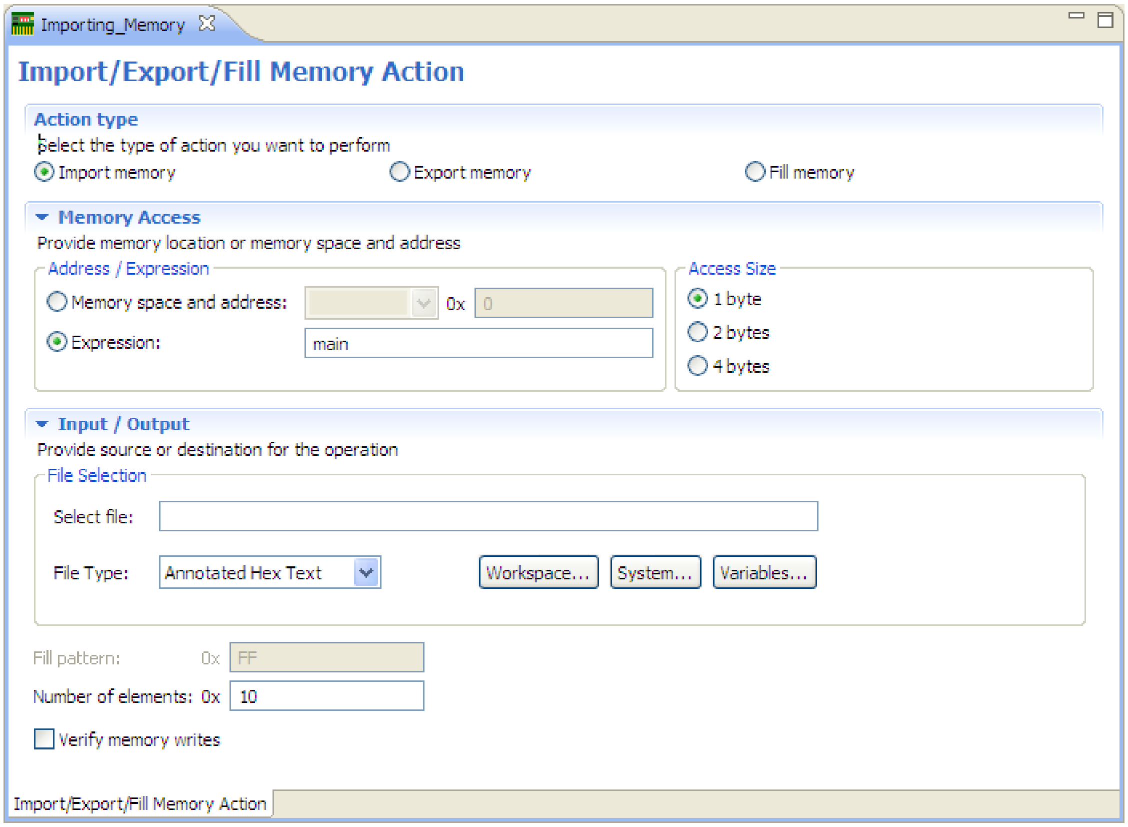
Figure 17-50. Create New Target Task Wizard

7. Click **Finish**.

The **Import/Export/Fill Memory Action** editor appears. This editor lets you read encoded data from a user specified file, decode it, and copy it into a user specified memory range.

NOTE

The **Import memory** option is selected by default.



Importing_Memory

Import/Export/Fill Memory Action

Action type

Select the type of action you want to perform

● Import memory ○ Export memory ○ Fill memory

Memory Access

Provide memory location or memory space and address

Address / Expression

○ Memory space and address: 0x 0

● Expression: main

Access Size

● 1 byte

○ 2 bytes

○ 4 bytes

Input / Output

Provide source or destination for the operation

File Selection

Select file:

File Type: Annotated Hex Text

Fill pattern: 0x FF

Number of elements: 0x 10

☐ Verify memory writes

Figure 17-51. Importing Memory

8. Specify options as explained in the following table.

NOTE

CodeWarrior IDE validates information as you enter it. If there are errors, a message appears near the page title.

Table 17-4. Import Data from a File into memory Page Options

Item	Description
Memory space and address	Enter the literal address and memory space on which the data transfer is performed. The Literal address field allows only decimal and hexadecimal values.
Expression	Enter the memory address or expression at which the data transfer starts.

Table continues on the next page...

Table 17-4. Import Data from a File into memory Page Options (continued)

Item	Description
Access size	Denotes the number of addressable units of memory that the debugger accesses in transferring one data element. The default values shown are 1, 2, 4, and 8 units. When target information is available, this list shall be filtered to display the access sizes that are supported by the target.
Select File	Enter the path to the file that contains the data to be imported. Click the **Browse** button to select the import file through the standard **File Open** dialog box.
File type	Defines the format in which the wizard encodes the data it imports. By default, the following file types are supported: • Annotated Hex Text • Hex Text • Motorola S-Record • Raw Binary • Signed Decimal Text • Unsigned decimal Text
Fill Pattern	Denotes the sequence of bytes, ordered from low to high memory, the wizard mirrors in the target. The field accept only hexadecimal values. If the width of the pattern exceeds the access size, the wizard displays an error message.
Number of Elements	Enter the total number of elements to be transferred.
Verify Memory Writes	Check the option to verify success of each data write to the memory.

9. Click the save icon on the IDE toolbar.

   CodeWarrior IDE saves your changes, closes the **Configure Import/Export/Fill Memory task** wizard, and displays the newly created import task in the **Tasks** list of the **Target Tasks** view.

10. Click the **Execute** icon to execute the task.

17.4.2 Creating Target Task to Export Memory

Perform these steps to create a target task to export memory:

1. Select **Window** > **Show View** > **Other**.

The **Show View** dialog box appears.

2. From the **Debug** group, select **Target Tasks**.

   The **Target Tasks** view appears.

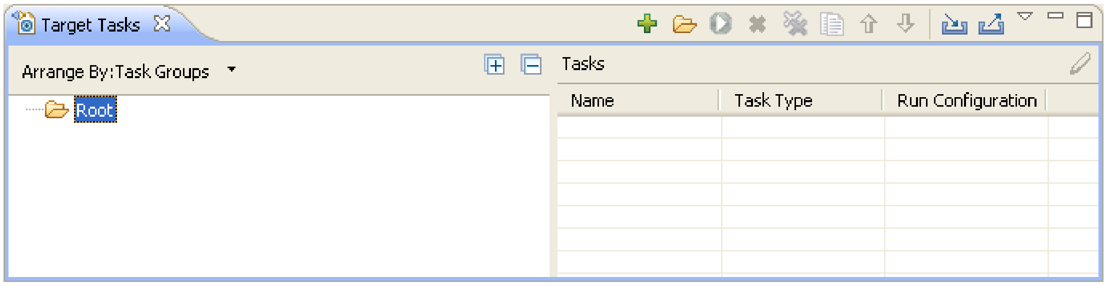


Figure 17-52. Target Tasks View

3. Right-click in the **Target Tasks** view and select **New Task** from the context menu.

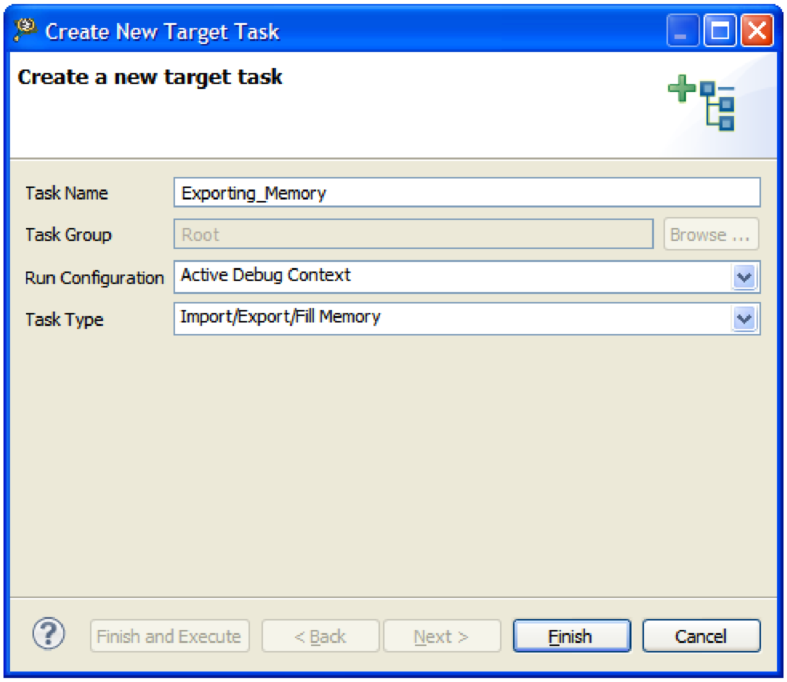
   The **Create New Target Task** wizard appears.

4. In the **Task Name** text box, enter a name for the new task. For example, `Exporting_Memory`.
5. Use the **Run Configuration** list box to specify the configuration that the task launches and uses to connect the target. For example, Active Debug Context.

NOTE

If the task does not successfully launch the configuration that you specify, the **Execute** button of the **Target Tasks** view toolbar stays disabled.

6. From the **Task Type** list box, select **Import/Export/Fill Memory**.



Task Name: Exporting_Memory

Task Group: Root

Run Configuration: Active Debug Context

Task Type: Import/Export/Fill Memory

Figure 17-53. Create New Target Task Wizard

7. Click **Finish**.

   The **Import/Export/Fill Memory Action** editor appears. This page lets you read data from a user specified memory range, encode it in a user specified format, and store this encoded data in a user specified output file.

8. Select the **Export memory** option.

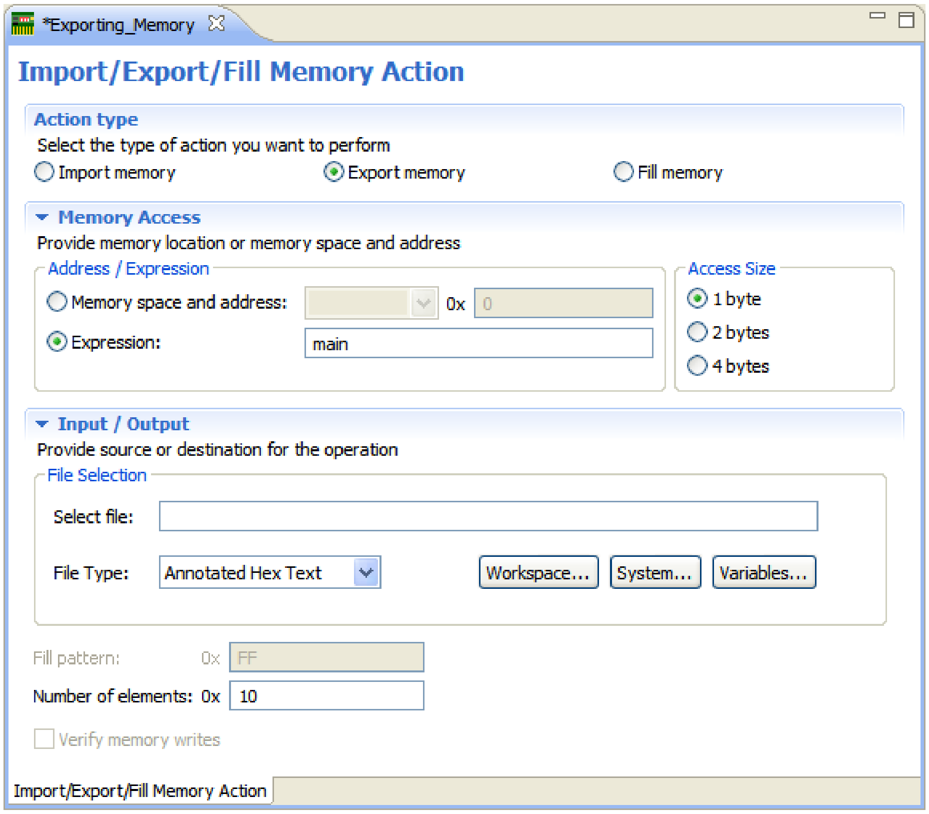

*Exporting_Memory
Import/Export/Fill Memory Action
Action type
Select the type of action you want to perform
○ Import memory
● Export memory
○ Fill memory
Memory Access
Provide memory location or memory space and address
Address / Expression
○ Memory space and address:
0x 0
● Expression: main
Access Size
● 1 byte
○ 2 bytes
○ 4 bytes
Input / Output
Provide source or destination for the operation
File Selection
Select file:
File Type: Annotated Hex Text
Workspace...
System...
Variables...
Fill pattern: 0x FF
Number of elements: 0x 10
□ Verify memory writes
Import/Export/Fill Memory Action

Figure 17-54. Exporting Memory

9. Specify options as explained in the following table.

NOTE

CodeWarrior IDE validates information as you enter it. If there are errors, a message appears near the page title.

Table 17-5. Export Data from Memory into a File Options

Item	Description
Memory space and address	Enter the literal address and memory space on which the data transfer is performed. The Literal address field allows only decimal and hexadecimal values.
Expression	Enter the memory address or expression at which the data transfer starts.
Access size	Denotes the number of addressable units of memory that the debugger accesses in transferring one data element. The default values shown are 1, 2, 4, and 8 units. When target information is available, this list shall be filtered to display the access sizes that are supported by the target.

Table continues on the next page...

Table 17-5. Export Data from Memory into a File Options (continued)

Item	Description
Select File	Enter the path to the file that contains the data to be imported. Click the **Browse** button to select the import file through the standard **File Open** dialog box.
File type	Defines the format in which the wizard encodes the data it imports. By default, the following file types are supported: • Annotated Hex Text • Hex Text • Motorola S-Record • Raw Binary • Signed Decimal Text • Unsigned decimal Text
Number of Elements	Enter the total number of elements to be transferred.

10. Click the save icon on the IDE toolbar.

    CodeWarrior IDE saves your changes, closes the **Configure Import/Export/Fill Memory task** wizard, and displays the newly created export task in the **Tasks** list of the **Target Tasks** view.

11. Click the **Execute** icon to execute the task.

17.4.3 Fill Memory with Data Pattern

To fill memory with a specified data pattern:

1. Select **Window > Show View > Other**.

   The **Show View** dialog box appears.

2. From the **Debug** group, select **Target Tasks**.

   The **Target Tasks** view appears.

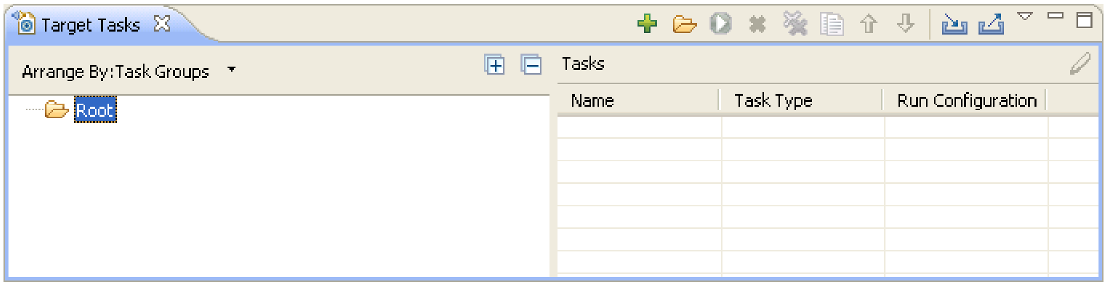


Figure 17-55. Target Tasks View

3. Right-click in the **Target Tasks** view and select **New Task** from the context menu.

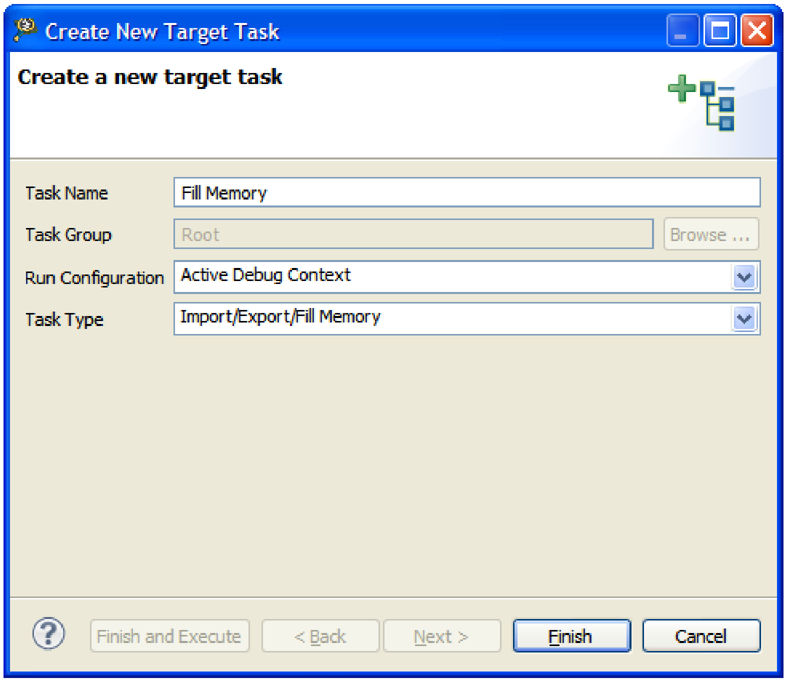
   The **Create New Target Task** wizard appears.

4. In the **Task Name** text box, enter a name for the new task. For example, *Fill Memory*.
5. Use the **Run Configuration** list box to specify the configuration that the task launches and uses to connect the target. For example, Active Debug Context.

NOTE

> If the task does not successfully launch the configuration that you specify, the **Execute** button of the **Target Tasks** view toolbar stays disabled.

6. From the **Task Type** list box, select **Import/Export/Fill Memory**.


Create New Target Task

Create a new target task

Task Name: Fill Memory

Task Group: Root

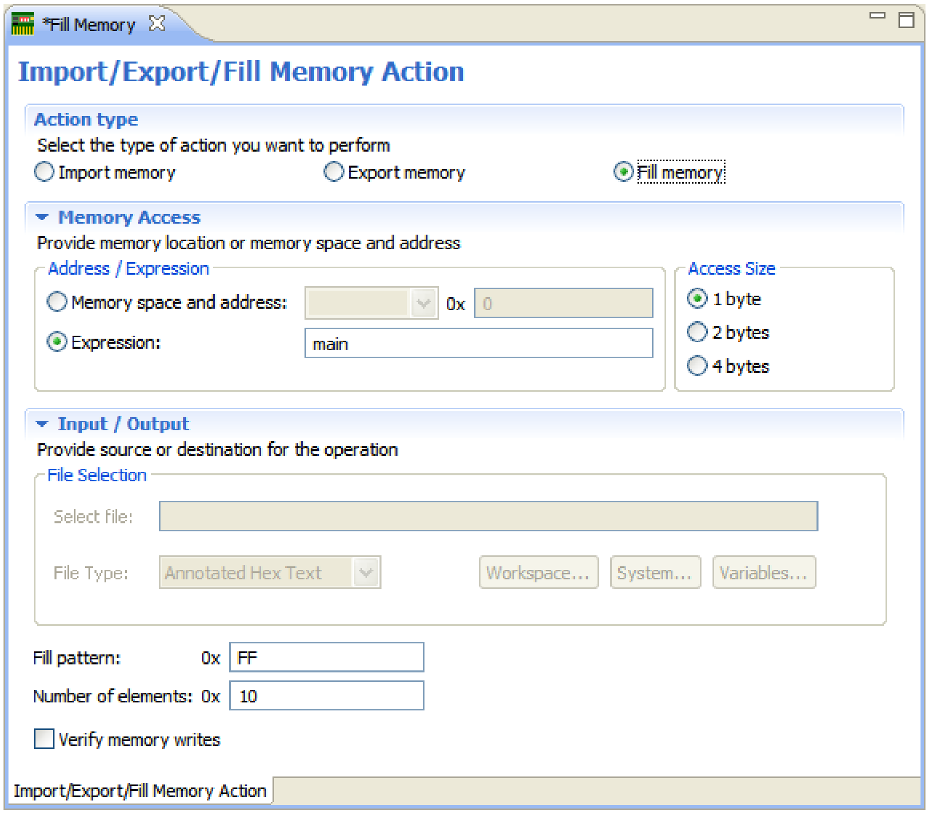
Run Configuration: Active Debug Context

Task Type: Import/Export/Fill Memory

Figure 17-56. Create New Target Task Wizard

7. Click **Finish.**
8. Select the **Fill memory** option.

   The **Import/Export/Fill Memory Action** editor appears. This page lets you fill a user specified memory range with a user specified data pattern.



*Fill Memory

Import/Export/Fill Memory Action

Action type

Select the type of action you want to perform

○ Import memory ○ Export memory ● Fill memory

Memory Access

Provide memory location or memory space and address

Address / Expression

○ Memory space and address: 0x 0

● Expression: main

Access Size

● 1 byte
○ 2 bytes
○ 4 bytes

Input / Output

Provide source or destination for the operation

File Selection

Select file:

File Type: Annotated Hex Text

Fill pattern: 0x FF

Number of elements: 0x 10

□ Verify memory writes

Import/Export/Fill Memory Action

Figure 17-57. Fill memory with a data pattern Page

9. Specify options as explained in the following table.

NOTE

CodeWarrior IDE validates information as you enter it. If there are errors, a message appears near the page title.

Table 17-6. Fill memory with a data pattern Page Options

Item	Description
Memory space and address	Enter the literal address and memory space on which the data transfer is performed. The Literal address field allows only decimal and hexadecimal values.
Expression	Enter the memory address or expression at which the data transfer starts.
Access size	Denotes the number of addressable units of memory that the debugger accesses in transferring one data element. The default values shown are 1, 2, 4, and 8 units. When target information is available, this list shall be filtered to display the access sizes that are supported by the target.

Table continues on the next page...

Table 17-6. Fill memory with a data pattern Page Options (continued)

Item	Description
Fill Pattern	Denotes the sequence of bytes, ordered from low to high memory, the wizard mirrors in the target. The field accept only hexadecimal values. If the width of the pattern exceeds the access size, the wizard displays an error message.
Number of Elements	Enter the total number of elements to be transferred.
Verify Memory Writes	Check this option to verify success of each data write to the memory.

10. Click the **Save** icon on the IDE toolbar.

    CodeWarrior IDE saves your changes, closes the **Configure Import/Export Memory task** wizard, and displays the newly created fill task in the **Tasks** list of the **TargetTasks** view.

11. Click the **Execute** icon to execute the task.

Chapter 18
CRC Utility for All Architectures

The Cyclic Redundancy Check or CRC is a technique, which is designed to check the integrity of user-defined memory areas. It helps in calculating the checksum of a binary application file.

Command Syntax

```
crcgen input -crc file [-vaddr] [-o file] [-srec [file] [length]] [-bin [file] [length]]
```

Listing: Input crc file syntax for crcgen utility

```
CRC
SEED = default_or_initial_value_of_checksum
FILL = 0xff
FROM start_address TO end_address;
FROM start_address TO end_address;
FROM start_address TO end_address;
DEST = destination_address
```

CRCgen.exe utility computes the single checksum value, CRC-16 (2-byte size) for the bytes that are present in all specified memory ranges (FROM start_address TO end_address) and stores the resulted checksum value in the destination address (DEST = destination_address).

It loads the ELF segment and calculates the checksum value and these values are written as distinct ELF segment.

Listing: Output format

```
**Program Segment **
Destination_address: checksum value
```

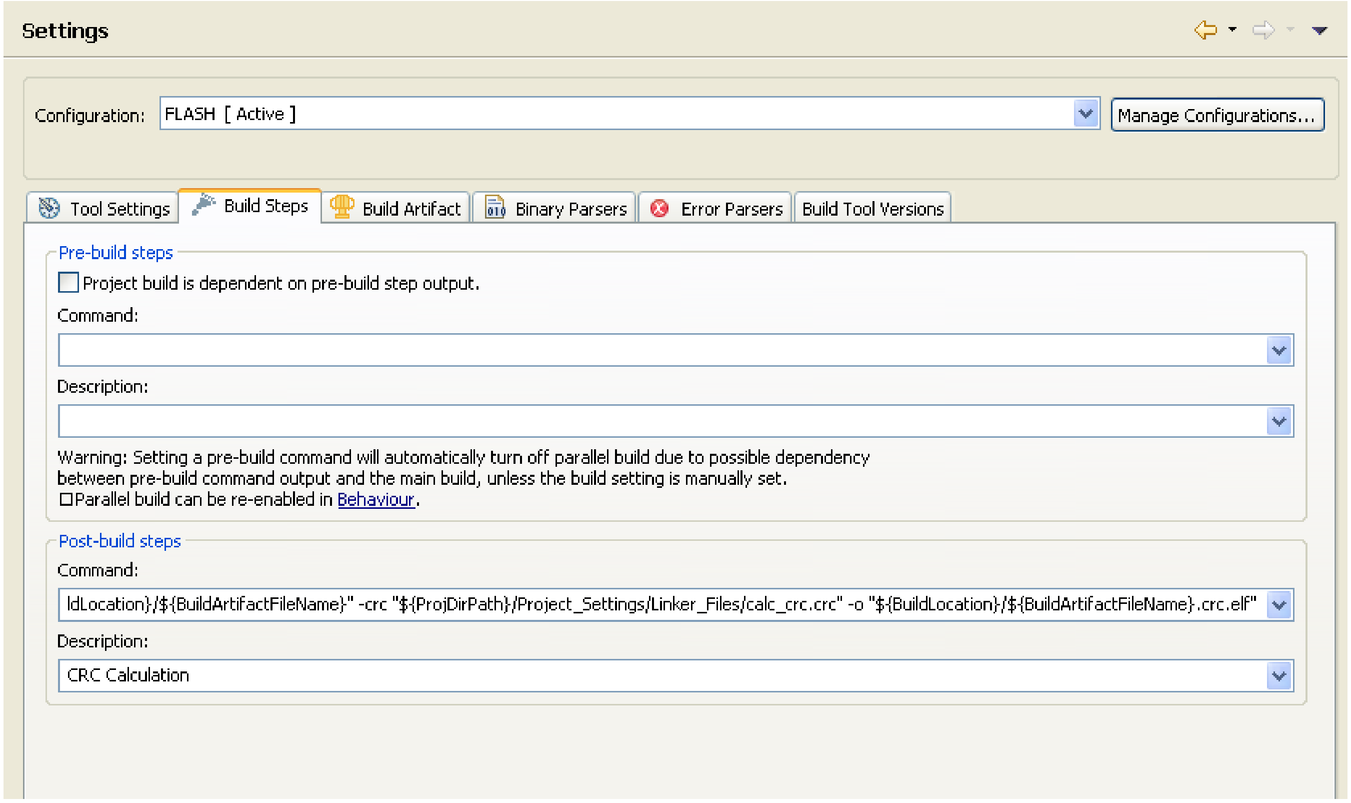
18.1 Using CRCgen on Microcontrollers

To use the CRCgen utility, perform the followings steps.

1. Create a file `calc_crc.crc` in the `Project/Project_Settings/Linker_Files` directory.
2. Configure the post-linker to use it.

Listing: Command line for post build

```
"${MCU_TOOLS_HOME}/bin/crcgen.exe" "${BuildLocation}/
${BuildArtifactFileName}" -crc "${ProjDirPath}/Project_Settings/
Linker_Files/calc_crc.crc" -o "${BuildLocation}/
${BuildArtifactFileName}.crc.elf"
```



Configuration: FLASH [Active]

Pre-build steps

□ Project build is dependent on pre-build step output.

Command:

Description:

Warning: Setting a pre-build command will automatically turn off parallel build due to possible dependency between pre-build command output and the main build, unless the build setting is manually set.
□Parallel build can be re-enabled in Behaviour.

Post-build steps

Command:

```
ldLocation}/${BuildArtifactFileName}" -crc "${ProjDirPath}/Project_Settings/Linker_Files/calc_crc.crc" -o "${BuildLocation}/${BuildArtifactFileName}.crc.elf"
```

Description:

CRC Calculation

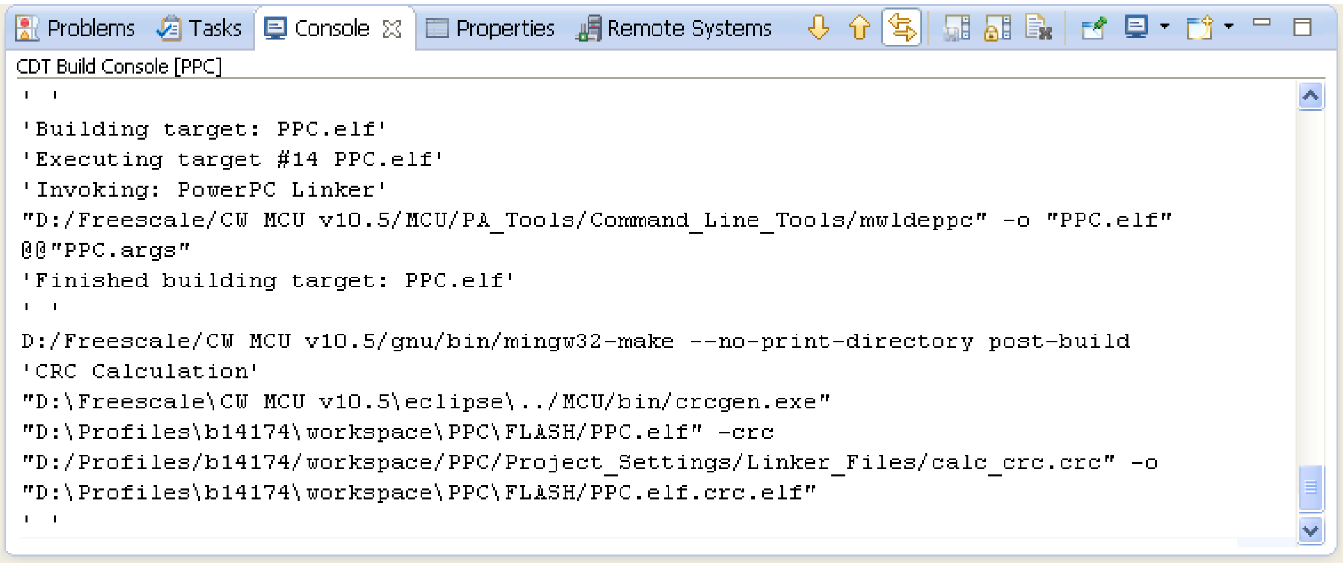
Figure 18-1. Post-build Steps in Microcontrollers

This command line uses several Eclipse variables:

- **${MCU_TOOLS_HOME}** : Points to the MCU folder inside the CodeWarrior for Microcontrollers v10.x installation.
- **$(BuildLocation}** : Points to the output folder inside the project of build where the ELF file is located.
- **${BuildArtifactFileName}** : Points to the variable that contains the ELF file.
- **${ProjDirPath}:**Points to the project folder.

3. Build the project.

   The post build step is executed and CRC of the application is calculated.



```
' '
'Building target: PPC.elf'
'Executing target #14 PPC.elf'
'Invoking: PowerPC Linker'
"D:/Freescale/CW MCU v10.5/MCU/PA_Tools/Command_Line_Tools/mwldeppc" -o "PPC.elf"
@@"PPC.args"
'Finished building target: PPC.elf'
' '
D:/Freescale/CW MCU v10.5/gnu/bin/mingw32-make --no-print-directory post-build
'CRC Calculation'
"D:\Freescale\CW MCU v10.5\eclipse\../MCU/bin/crcgen.exe"
"D:\Profiles\b14174\workspace\PPC\FLASH/PPC.elf" -crc
"D:/Profiles/b14174/workspace/PPC/Project_Settings/Linker_Files/calc_crc.crc" -o
"D:\Profiles\b14174\workspace\PPC\FLASH/PPC.elf.crc.elf"
' '
```

Figure 18-2. Post Build Step in Console View

18.2 Examples

The CRC command, the CRC file syntax and the options are generic and work for all the architectures and devices.

The following illustrate examples on specific architecture and device.

18.2.1 ColdFire

Target Family : MCF5485

Listing: Memory Segments

```
MEMORY {
   vectorram    (RWX)  : ORIGIN = 0x20000000, LENGTH = 0x00000400
   userram      (RWX)  : ORIGIN = 0x20000400, LENGTH = 0x00001A00
   code         (RX)   : ORIGIN = 0x10010000, LENGTH = 0x00008000
}
```

Listing: Input calc_crc.crc File

```
CRC
SEED = 0xFEF
FILL = 0xFF
FROM 0x10010000 TO 0x10010004;    //range from code segment
DEST = 0x10017FFA
```

Listing: Input Bytes from .elf File

```
                         *** PROGRAM SEGMENT 7 ***
0x10010000:  20 00 1F FC 10     //memory range 0x10010000 To 0x10010004
```

Listing: Program Header Before Post Linking

```
                        *** PROGRAM HEADER TABLE ***
no type offset     vaddr      paddr      filesz     memsz      flags     align

0  LOAD 0x00000154 0x20000000 0x20000000 0x00000000 0x00000000 0x00000007 4
1  LOAD 0x00000154 0x20000400 0x20000400 0x00000000 0x00000000 0x00000007 4
2  LOAD 0x00003674 0x20000400 0x10013520 0x00000018 0x00000018 0x00000007 4
3  LOAD 0x0000368C 0x20000418 0x20000418 0x00000000 0x00000004 0x00000007 4
4  LOAD 0x0000368C 0x2000041C 0x2000041C 0x00000000 0x00000000 0x00000007 4
5  LOAD 0x0000368C 0x2000041C 0x10013538 0x00000018 0x00000018 0x00000007 4
6  LOAD 0x00000154 0x10010000 0x10010000 0x00000000 0x00000000 0x00000005 4
7  LOAD 0x00000154 0x10010000 0x10010000 0x00000400 0x00000400 0x00000005 4
8  LOAD 0x00000554 0x10010400 0x10010400 0x00003120 0x00003120 0x00000005 4
```

Listing: Output in '.elf.crc.elf' file

```
                         *** PROGRAM SEGMENT 9 ***
0x10017FFA: 16 86      //calculated checksum value -2 bytes
```

Listing: Program Header after post linking

```
*** PROGRAM HEADER TABLE ***
no type offset     vaddr      paddr      filesz     memsz      flags     align

0  LOAD 0x00000174 0x20000000 0x20000000 0x00000000 0x00000000 0x00000007  4
1  LOAD 0x00000174 0x20000400 0x20000400 0x00000000 0x00000000 0x00000007  4
2  LOAD 0x00003694 0x20000400 0x10013520 0x00000018 0x00000018 0x00000007  4
3  LOAD 0x000036AC 0x20000418 0x20000418 0x00000000 0x00000004 0x00000007  4
4  LOAD 0x000036AC 0x2000041C 0x2000041C 0x00000000 0x00000000 0x00000007  4
5  LOAD 0x000036AC 0x2000041C 0x10013538 0x00000018 0x00000018 0x00000007  4
6  LOAD 0x00000174 0x10010000 0x10010000 0x00000000 0x00000000 0x00000005  4
7  LOAD 0x00000174 0x10010000 0x10010000 0x00000400 0x00000400 0x00000005  4
8  LOAD 0x00000574 0x10010400 0x10010400 0x00003120 0x00003120 0x00000005  4
9  LOAD 0x0000F864 0x10017FFA 0x10017FFA 0x00000002 0x00000002 0x00000004  0
```

New segment added by the post linker or crcgen utility.

18.2.2 PowerPC

Target Family : MPC5675K

Listing: Memory Segments

```
MEMORY
{
   /* FLASH: 0x00000000 - 0x001FFFFF */
resetvector:     org = 0x00000000,  len = 0x00000010
init:            org = 0x00000010,  len = 0x00000FF0 /* ~4K */
exception_handlers_p0: org = 0x00010000, len = 0x00010000 /* 64K core_0 */
exception_handlers_p1: org = 0x00020000, len = 0x00020000 /* 128K core_1 */
internal_flash:         org = 0x00040000, len = 0x001C0000 /* 1792 KB */
```

Listing: Input calc_crc.crc File

```
CRC
SEED = 0xFFFF
FILL = 0xFF
FROM 0x00040000 TO 0x00040008;     //range from internal_flash segment
DEST = 0x1FFFEA
```

To read bytes present at specified memory location go to file offset.

Listing: Input Bytes from .elf file:

```
Program Header [7]
 p_type:   00000001, p_offset: 00000e00
p_vaddr:  00040000, p_paddr:  00000000
 p_filesz: 0000027c, p_memsz:  0000027c, p_flags: 10000005, p_align: 00000010


E00: 70 08 E0 00 70 00 C0 00 70     //memory range from 0x00040000 To 0x00040008
```

Listing: Program Header Before Post Linking

```
Program Header [0]
  p_type:   00000001, p_offset: 00000360
  p_vaddr:  00000000, p_paddr:  00000000
  p_filesz: 00000008, p_memsz:  00000008, p_flags: 00000004, p_align: 00000004
.
.
.
Program Header [7]
  p_type:   00000001, p_offset: 00000e00
  p_vaddr:  00040000, p_paddr:  00000000
  p_filesz: 0000027c, p_memsz:  0000027c, p_flags: 10000005, p_align: 00000010
.
.
.
Program Header [13]
  p_type:   00000001, p_offset: 00002028
  p_vaddr:  40001528, p_paddr:  00000000
  p_filesz: 00000000, p_memsz:  0000000c, p_flags: 00000006, p_align: 00000008

Program Header [14]
  p_type:   00000001, p_offset: 00003000
  p_vaddr:  50000000, p_paddr:  00000000
  p_filesz: 00000000, p_memsz:  00001500, p_flags: 00000006, p_align: 00001000
```

To read the checksum value go to the file offset.

Listing: Output in '.elf.crc.elf' File

```
Program Header [15]
 p_type:   00000001, p_offset: 00009cb8
 p_vaddr:  001fffea, p_paddr:  001fffea
 p_filesz: 00000002, p_memsz:  00000002, p_flags: 00000004, p_align: 00000000


9CB0: 00 00 00 10 00 00 00 00 F0 76   //calculated checksum value -2 bytes
```

Listing: Program Header After Post Linking

```
Program Header [0]
 p_type:   00000001, p_offset: 00000360
 p_vaddr:  00000000, p_paddr:  00000000
 p_filesz: 00000008, p_memsz:  00000008, p_flags: 00000004, p_align: 00000004
.
.
.
Program Header [7]
 p_type:   00000001, p_offset: 00000e00
 p_vaddr:  00040000, p_paddr:  00000000
 p_filesz: 0000027c, p_memsz:  0000027c, p_flags: 10000005, p_align: 00000010
.
.
.
```

```
Program Header [13]
 p_type:   00000001, p_offset: 00002028
 p_vaddr:  40001528, p_paddr:  00000000
 p_filesz: 00000000, p_memsz:  0000000c, p_flags: 00000006, p_align: 00000008

Program Header [14]
 p_type:   00000001, p_offset: 00003000
 p_vaddr:  50000000, p_paddr:  00000000
 p_filesz: 00000000, p_memsz:  00001500, p_flags: 00000006, p_align: 00001000

Program Header [15]
 p_type:   00000001, p_offset: 00009cb8
 p_vaddr:  001fffea, p_paddr:  001fffea
 p_filesz: 00000002, p_memsz:  00000002, p_flags: 00000004, p_align: 00000000
```

New segment added by the post linker or crcgen utility.

18.2.3 Freescale ARM

Target Family : PK60N512

Listing: Memory Segments

```
MEMORY
{
m_interrupts  (RX) : ORIGIN = 0x00000000, LENGTH = 0x000001E0
m_text        (RX) : ORIGIN = 0x00000800, LENGTH = 0x0007F800
m_data        (RW) : ORIGIN = 0x1FFF0000, LENGTH = 0x00020000
m_cfmprotrom  (RX) : ORIGIN = 0x00000400, LENGTH = 0x00000010
}
```

Listing: Input calc_crc.crc File

```
CRC
SEED = 0xfef
FILL = 0xff
FROM 0x00000800 TO 0x00000804;         //ranges from m_text segment
DEST = 0x7F400
```

Listing: Input Bytes from .elf File

```
  *** PROGRAM SEGMENT 1 ***
0x00000800: DF F8 34 10 4E   //for memory range from 0x00000800 To 0x00000804
```

Listing: Program Header Before Post Linking

```
                           *** PROGRAM HEADER TABLE ***
no type offset     vaddr      paddr      filesz     memsz      flags     align
0  LOAD 0x000000F4 0x00000000 0x00000000 0x000001E0 0x000001E0 0x00000005  4
1  LOAD 0x000002D8 0x00000800 0x00000800 0x00000D84 0x00000D84 0x80000005  8
2  LOAD 0x00001060 0x1FFF0000 0x00001584 0x000000D8 0x000000D8 0x00000006  8
3  LOAD 0x00001138 0x1FFF00D8 0x1FFF00D8 0x00000000 0x000000E8 0x00000006  8
4  LOAD 0x00001138 0x1FFF01C0 0x0000165C 0x00000018 0x00000018 0x00000006  4
5  LOAD 0x000002D4 0x00000400 0x00000400 0x00000000 0x00000000 0x00000005  4
```

Listing: Output in '.elf.crc.elf' file

```
                       *** PROGRAM SEGMENT 6 ***
0x0007F400:  74 CE            //calculated checksum value -2 bytes
```

Listing: Program Header After Post Linking

```
                             *** PROGRAM HEADER TABLE ***
no type offset     vaddr      paddr      filesz     memsz      flags     align
0  LOAD 0x00000114 0x00000000 0x00000000 0x000001E0 0x000001E0 0x00000005  4
1  LOAD 0x000002F8 0x00000800 0x00000800 0x00000D84 0x00000D84 0x80000005  8
2  LOAD 0x00001080 0x1FFF0000 0x00001584 0x000000D8 0x000000D8 0x00000006  8
3  LOAD 0x00001158 0x1FFF00D8 0x1FFF00D8 0x00000000 0x000000E8 0x00000006  8
4  LOAD 0x00001158 0x1FFF01C0 0x0000165C 0x00000018 0x00000018 0x00000006  4
5  LOAD 0x000002F4 0x00000400 0x00000400 0x00000000 0x00000000 0x00000005  4
6  LOAD 0x0000A444 0x0007F400 0x0007F400 0x00000002 0x00000002 0x00000004  0
```

New segment added by the post linker or crcgen utility.

18.2.4 GCC ARM

Target Family: MK60DN512Z

Listing: Memory Segments

```
MEMORY
{

 m_interrupts (rx) : ORIGIN = 0x00000000, LENGTH = 0x1E0
 m_cfmprotrom (rx) : ORIGIN = 0x00000400, LENGTH = 0x10
 m_text (rx) : ORIGIN = 0x00000800, LENGTH = 512K - 0x800  m_data    (rwx) : ORIGIN =
0x1FFF0000, LENGTH = 64K /* Lower SRAM */
  m_data2   (rwx) : ORIGIN = 0x20000000, LENGTH = 64K  /* Upper SRAM */
}
```

Listing: Input calc_crc.crc file

```
CRC
SEED = 0xFEF
FILL = 0xFF
FROM 0x00000800 TO 0x00000804; //memory range from m_text segment
DEST = 0x7FFEA
```

To read bytes present at specified memory location go to file offset.

Listing: Input Bytes from .elf file

```
LOAD off 0x000002b4 vaddr 0x00000800 paddr 0x00000800 align 2**2
     filesz 0x000013f0 memsz 0x000013f0 flags r-x


2B0: 01 08 00 00 80 B4 00 AF 00 //bytes present in memory range from 0x00000800 TO 0x00000804
```

Listing: Program Header Before Post Linking

```
Program Header:
 LOAD off 0x000000d4 vaddr 0x00000000 paddr 0x00000000 align 2**2
   filesz 0x000001e0 memsz 0x000001e0 flags r--
 LOAD off    0x000002b4 vaddr 0x00000800 paddr 0x00000800 align 2**2
   filesz 0x000013f0 memsz 0x000013f0 flags r-x
 LOAD off    0x000016a4 vaddr 0x1fff0000 paddr 0x00001bf0 align 2**2
   filesz 0x00000014 memsz 0x00000030 flags rw-
 LOAD off    0x000016b8 vaddr 0x20000000 paddr 0x00001c04 align 2**0
   filesz 0x00000024 memsz 0x00000024 flags rw-
 LOAD off    0x000016dc vaddr 0x1fff0030 paddr 0x00001c20 align 2**0
   filesz 0x00000000 memsz 0x00001000 flags rw-
```

To read the checksum value go to the file offset.

Listing: Output in .elf.crc.elf File

```
LOAD off    0x000777bc vaddr 0x0007ffea paddr 0x0007ffea align 2**0
         filesz 0x00000002 memsz 0x00000002 flags r--


777B0: 5F 64 61 74 61 5F 73 69 7A 65 00 00 4D 21 //calculated checksum value -2 bytes
```

Listing: Program Header After Post Linking

```
Program Header:
  LOAD off    0x000000f4 vaddr 0x00000000 paddr 0x00000000 align 2**2
    filesz 0x000001e0 memsz 0x000001e0 flags r--
  LOAD off    0x000002d4 vaddr 0x00000800 paddr 0x00000800 align 2**2
    filesz 0x000013f0 memsz 0x000013f0 flags r-x
  LOAD off    0x000016c4 vaddr 0x1fff0000 paddr 0x00001bf0 align 2**2
    filesz 0x00000014 memsz 0x00000030 flags rw-
  LOAD off    0x000016d8 vaddr 0x20000000 paddr 0x00001c04 align 2**0
    filesz 0x00000024 memsz 0x00000024 flags rw-
  LOAD off    0x000016fc vaddr 0x1fff0030 paddr 0x00001c20 align 2**0
    filesz 0x00000000 memsz 0x00001000 flags rw-
 LOAD off 0x000777bc vaddr 0x0007ffea paddr 0x0007ffea align 2**0 filesz 0x00000002 memsz
0x00000002 flags r--
```

New segment added by the post linker or crcgen utility.

18.2.5 DSC

Target Family : MC56F84789

Listing: Memory Segments

```
MEMORY {
## Program Memory space
.p_interruptsboot_ROM(RX) : ORIGIN = 0x000000, LENGTH = 0x000004 # reserved for boot
location
.p_interrupts_ROM(RX)  : ORIGIN = 0x000004,  LENGTH = 0x000200 # reserved for interrupt
vectoring
# .p_reserved_IFR     (RX)  : ORIGIN = 0x000200, LENGTH = 0x000008 # 16 bytes reserved for
IFR
.p_flash_ROM          (RX)  : ORIGIN = 0x000208, LENGTH = 0x01FDF8 # primary location for
code to be run - to 0x1ffff
```

Listing: Input calc_crc.crc File

```
CRC
SEED = 0xFFFF
FILL = 0xFF
FROM 0x000208 TO 0x000210;
DEST = 0x3000            //Address as DATA BYTE
```

Listing: Input Bytes from .elf File

```
*** PROGRAM SEGMENT 2 ***
0x00000208:  7B 82 4B 86 0A 00 7B 82 48 87 00 40 1F D8 7C F8  '{.K...{.H..@..|.'
0x00000210:  BC
```

Listing: Program Header Before Post Linking

```
*** PROGRAM HEADER TABLE ***
no type offset     vaddr      paddr      filesz     memsz      flags     align
```

```
0  LOAD 0x000001B4 0x00000000 0x00000000 0x00000008 0x00000008 0x00000005  1
1  LOAD 0x000001BC 0x00000004 0x00000004 0x000001B4 0x000001B4 0x00000005  1
2  LOAD 0x00000370 0x00000208 0x00000208 0x000039BC 0x000039BC 0x00000005  4
3  LOAD 0x00000000 0x00068000 0x00068000 0x00000000 0x00008000 0x00000005  1
4  LOAD 0x00003D2C 0x00001EE6 0x00002000 0x00000290 0x00000290 0x00000005  4
5  LOAD 0x00000000 0x00000000 0x00000000 0x00000000 0x00004000 0x00000006  1
6  LOAD 0x00003FC0 0x00002000 0x00002000 0x00000290 0x00000DB0 0x00000006  8
7  LOAD 0x00000000 0x0000C000 0x0000C000 0x00000000 0x00004000 0x00000006  1
8  LOAD 0x00000000 0x0000E000 0x0000E000 0x00000000 0x00004000 0x00000006  1
9  LOAD 0x00000000 0x0001E000 0x0001E000 0x00000000 0x00000800 0x00000006  1
10 LOAD 0x00000000 0x0001E400 0x0001E400 0x00000000 0x00003800 0x00000006  1
11 LOAD 0x00000000 0x00FFFF00 0x00FFFF00 0x00000000 0x00000200 0x00000006  1
```

Listing: Output in '.elf.crc.elf' file

```
                              ** PROGRAM SEGMENT 12 ***
0x00001800:  56 D0    //
Address as "DATA WORD"
--calculated checksum value -2 bytes
```

NOTE

DEST (destination_address) in the input ***.crc** file for DSC should be a **BYTE ADDRESS** value. In the above example, ' *DEST = 0x3000*' in the ' *calc_crc.crc*' file.

Since the DSC disassembler output lists the segments addresses as **WORD ADDRESS** , the checksum value segment in the final disassembly should be checked using the corresponding **WORD ADDRESS** .

For DSC, the byte address is converted to a word address by dividing the byte address by two. For example, the checksum data value generated for the above example is located at *0x1800 'WORD ADDRESS'*.

Listing: Program Header after post linking

```
                              *** PROGRAM HEADER TABLE ***
no type offset      vaddr       paddr       filesz      memsz       flags      align

0  LOAD 0x000001D4  0x00000000  0x00000000  0x00000008  0x00000008  0x00000005  1
1  LOAD 0x000001DC  0x00000004  0x00000004  0x000001B4  0x000001B4  0x00000005  1
2  LOAD 0x00000390  0x00000208  0x00000208  0x000039BC  0x000039BC  0x00000005  4
3  LOAD 0x00000000  0x00068000  0x00068000  0x00000000  0x00008000  0x00000005  1
4  LOAD 0x00003D4C  0x00001EE6  0x00002000  0x00000290  0x00000290  0x00000005  4
5  LOAD 0x00000000  0x00000000  0x00000000  0x00000000  0x00004000  0x00000006  1
6  LOAD 0x00003FE0  0x00002000  0x00002000  0x00000290  0x00000DB0  0x00000006  8
7  LOAD 0x00000000  0x0000C000  0x0000C000  0x00000000  0x00004000  0x00000006  1
8  LOAD 0x00000000  0x0000E000  0x0000E000  0x00000000  0x00004000  0x00000006  1
9  LOAD 0x00000000  0x0001E000  0x0001E000  0x00000000  0x00000800  0x00000006  1
10 LOAD 0x00000000  0x0001E400  0x0001E400  0x00000000  0x00003800  0x00000006  1
11 LOAD 0x00000000  0x00FFFF00  0x00FFFF00  0x00000000  0x00000200  0x00000006  1
12 LOAD 0x0001BC7C  0x00001800  0x00001800  0x00000002  0x00000002  0x00000004  0
```

New segment added by the post linker or crcgen utility.

18.2.6 8/16 bit

Target Family : MC9S08QE128

Listing: Memory Segments

```
SEGMENTS /* Here all RAM/ROM areas of the device are listed. Used in PLACEMENT below. */
    Z_RAM                    = READ_WRITE  0x0080 TO 0x00FF;
    RAM                      = READ_WRITE  0x0100 TO 0x17FF;
    RAM1                     = READ_WRITE  0x1880 TO 0x207F
 /* unbanked FLASH ROM */
    ROM = READ_ONLY 0x2080 TO 0x7FFF;
    ROM1                     = READ_ONLY   0xC000 TO 0xFFAD;
```

Listing: Input calc_crc.crc file

```
CRC
SEED = 0xfef
FILL = 0xff
FROM 0x211E TO 0x2123;
DEST = 0x21D2
```

To read bytes present at specified memory location go to file offset.

Listing: Input Bytes from .elf file

```
NO   TYPE    OFFSET SIZE VIRTADDR PHYADDR MEMSIZE FLAGS ALIGNMNT
28 - PT_LOAD D4       C5     211E       0      C5   R X       0


D0 : 00 00 00 00  A7 FA 45 0F EF 9E     //memory range from 0x211E TO 0x2123
```

Listing: Program Header Before Post Linking

```
PROGRAM HEADER TABLE - 32 Items
Starts at:    148D1, Size of an entry:     20, Ends at:    14CD1
  NO   TYPE         OFFSET     SIZE VIRTADDR  PHYADDR  MEMSIZE  FLAGS ALIGNMNT
   0 - PT_LOAD          34        0        0        0       2C   RW          0
   1 - PT_LOAD          34        0       2D        0        9   RW          0
   2 - PT_LOAD          34        0       38        0        7   RW          0
   3 - PT_LOAD          34        0       40        0        E   RW          0
   4 - PT_LOAD          34        0       50        0        E   RW          0
   5 - PT_LOAD          34        0       60        0       17   RW          0
   6 - PT_LOAD          34        0       78        0        8   RW          0
   7 - PT_LOAD          34        0      100        0      219   RW          0
   8 - PT_LOAD          34        0     1800        0        4   RW          0
   9 - PT_LOAD          34        0     1806        0        4   RW          0
  10 - PT_LOAD          34        0     180B        0        1   RW          0
  11 - PT_LOAD          34        0     180E        0       14   RW          0
  12 - PT_LOAD          34        0     1823        0        4   RW          0
  13 - PT_LOAD          34        0     1830        0        3   RW          0
  14 - PT_LOAD          34        0     1838        0        4   RW          0
  15 - PT_LOAD          34        0     183D        0        1   RW          0
  16 - PT_LOAD          34        0     1840        0        3   RW          0
  17 - PT_LOAD          34        0     1844        0        3   RW          0
  18 - PT_LOAD          34        0     1848        0        3   RW          0
  19 - PT_LOAD          34        0     184C        0        3   RW          0
  20 - PT_LOAD          34        0     1850        0        3   RW          0
  21 - PT_LOAD          34        0     1854        0        3   RW          0
  22 - PT_LOAD          34        0     1858        0        3   RW          0
  23 - PT_LOAD          34        0     185C        0        3   RW          0
  24 - PT_LOAD          34        0     1860        0        3   RW          0
  25 - PT_LOAD          34        0     1868        0        6   RW          0
  26 - PT_LOAD          34        0     1870        0        E   RW          0
  27 - PT_LOAD          34       9E     2080        0       9E   R X         0
  28 - PT_LOAD          D4       C5     211E        0       C5   R X         0
  29 - PT_LOAD         19C       1E     21E3        0       1E   R X         0
  30 - PT_LOAD         1BC        2     FFFE        0        2   R X         0
  31 - PT_NOTE       119EC      30D
```

To read the checksum value go to the file offset.

Listing: Output in '.elf.crc.elf' file

```
NO   TYPE    OFFSET     SIZE VIRTADDR PHYADDR MEMSIZE FLAGS ALIGNMNT
32 - PT_LOAD 14CF2        2     21D2     21D2        2   R       0


14CF0: 00 00 57 74          //calculated checksum value -2 bytes
```

Listing: Program Header After Post Linking

```
PROGRAM HEADER TABLE - 33 Items
Starts at:    148D1, Size of an entry:     20, Ends at:    14CF1
  NO   TYPE         OFFSET     SIZE VIRTADDR  PHYADDR  MEMSIZE  FLAGS ALIGNMNT
   0 - PT_LOAD          34        0        0        0       2C   RW          0
   1 - PT_LOAD          34        0       2D        0        9   RW          0
   2 - PT_LOAD          34        0       38        0        7   RW          0
   3 - PT_LOAD          34        0       40        0        E   RW          0
   4 - PT_LOAD          34        0       50        0        E   RW          0
   5 - PT_LOAD          34        0       60        0       17   RW          0
   6 - PT_LOAD          34        0       78        0        8   RW          0
   7 - PT_LOAD          34        0      100        0      219   RW          0
   8 - PT_LOAD          34        0     1800        0        4   RW          0
   9 - PT_LOAD          34        0     1806        0        4   RW          0
  10 - PT_LOAD          34        0     180B        0        1   RW          0
  11 - PT_LOAD          34        0     180E        0       14   RW          0
  12 - PT_LOAD          34        0     1823        0        4   RW          0
  13 - PT_LOAD          34        0     1830        0        3   RW          0
  14 - PT_LOAD          34        0     1838        0        4   RW          0
  15 - PT_LOAD          34        0     183D        0        1   RW          0
  16 - PT_LOAD          34        0     1840        0        3   RW          0
  17 - PT_LOAD          34        0     1844        0        3   RW          0
  18 - PT_LOAD          34        0     1848        0        3   RW          0
  19 - PT_LOAD          34        0     184C        0        3   RW          0
  20 - PT_LOAD          34        0     1850        0        3   RW          0
  21 - PT_LOAD          34        0     1854        0        3   RW          0
  22 - PT_LOAD          34        0     1858        0        3   RW          0
  23 - PT_LOAD          34        0     185C        0        3   RW          0
  24 - PT_LOAD          34        0     1860        0        3   RW          0
  25 - PT_LOAD          34        0     1868        0        6   RW          0
  26 - PT_LOAD          34        0     1870        0        E   RW          0
  27 - PT_LOAD          34       9E     2080        0       9E   R X         0
  28 - PT_LOAD          D4       C5     211E        0       C5   R X         0
  29 - PT_LOAD         19C       1E     21E3        0       1E   R X         0
  30 - PT_LOAD         1BC        2     FFFE        0        2   R X         0
  31 - PT_NOTE       119EC      30D
  32 - PT_LOAD       14CF2        2     21D2     21D2        2   R           0
```

New segment added by the post linker or crcgen.exe utility.

18.2.7 S12Z

Target Family: MC9S12ZVFP64

Listing: Memory Segments

```
SEGMENTS  /* Here all RAM/ROM areas of the device are listed. Used in
PLACEMENT below. */
/* Register space  */
/*    IO_SEG        = PAGED         0x000000 TO   0x000FFF;
intentionally not defined */
/* RAM */
      RAM           = READ_WRITE  0x001000 TO 0x001FFF;
/* EEPROM */
      EEPROM        = READ_ONLY   0x100000 TO 0x1007FF;
/* non-paged FLASHs */
      ROM           = READ_ONLY   0xFF0000 TO 0xFFFDFF;
```

Listing: Input calc_crc.crc file

```
CRC
SEED = 0xfef
FILL = 0xff
FROM  0xFF0000 TO 0xFF0004;
DEST = 0xFFFDFF
```

To read bytes present at specified memory location go to file offset.

Listing: Input Bytes from .elf file

```
NO   TYPE    OFFSET SIZE VIRTADDR PHYADDR MEMSIZE FLAGS ALIGNMNT
93 - PT_LOAD 34     5B   FF0000   0       5B      R X   0


30: 01 C6 01 C5 B6 FF 00 40 27 //memory range from 0xFF0000 TO 0xFF0004
```

Listing: Program Header before post linking

```
PROGRAM HEADER TABLE - 98 Items
Starts at:    28920, Size of an entry:     20, Ends at:    29560
NO   TYPE         OFFSET     SIZE VIRTADDR  PHYADDR  MEMSIZE  FLAGS ALIGNMNT
0 - PT_LOAD          34        0        0        0         4   RW          0
1 - PT_LOAD          34        0       10        0         2   RW          0
2 - PT_LOAD          34        0       17        0         9   RW          0
3 - PT_LOAD          34        0       70        0         1   RW          0
4 - PT_LOAD          34        0       80        0         4   RW          0
.
.
.
.
. 93 - PT_LOAD 34 5B FF0000 0 5B R X 0
94 - PT_LOAD           90       C9   FF005B        0       C9   R X         0
95 - PT_LOAD          15C       21   FF0124        0       21   R X         0
96 - PT_LOAD          180        3   FFFFFD        0        3   R X         0
97 - PT_NOTE       21934      2CF
```

To read the checksum value go to the file **offset**.

Listing: Output in '.elf.crc.elf' file

```
NO   TYPE         OFFSET     SIZE VIRTADDR PHYADDR MEMSIZE FLAGS ALIGNMNT
98 - PT_LOAD       29580        2   FFFDFF   FFFDFF     2   R        0


29580: C4 AC //calculated checksum value -2 bytes
```

Listing: Program Header after post linking

```
PROGRAM HEADER TABLE - 99 Items
Starts at:    28920, Size of an entry:     20, Ends at:    29580
```

```
NO   TYPE         OFFSET     SIZE VIRTADDR  PHYADDR  MEMSIZE  FLAGS ALIGNMNT
0 - PT_LOAD          34         0        0        0        4   RW          0
1 - PT_LOAD          34         0       10        0        2   RW          0
2 - PT_LOAD          34         0       17        0        9   RW          0
3 - PT_LOAD          34         0       70        0        1   RW          0
4 - PT_LOAD          34         0       80        0        4   RW          0
.
.
.
.
.
93 - PT_LOAD          34       5B   FF0000        0       5B   R X         0
94 - PT_LOAD          90       C9   FF005B        0       C9   R X         0
95 - PT_LOAD         15C       21   FF0124        0       21   R X         0
96 - PT_LOAD         180        3   FFFFFD        0        3   R X         0
97 - PT_NOTE       21934      2CF
98 - PT_LOAD       29580        2   FFFDFF   FFFDFF        2   R           0
```

New segment added by the post linker or crcgen.exe utility

18.3 Application Example

Quick example for ARM PK60N512.

```
#include<stdio.h>
#define CRC16START 0xFEF            // CRC16/Checksum Start Value (seed value according
to .crc file)
unsigned char CalcCRC16(void);

unsigned int crc16calc(unsigned short crc16, unsigned char data);
unsigned long
Memory[]={0x00000800,0x00000801,0x00000802,0x00000803,0x00000804}; //Array of addresses for
which we want to calculate checksum


unsigned char CalcCRC16(void)
{
unsigned short checked_data = CRC16START;       // Start value of CRC16/Check-Sum

unsigned char data;                             // current data to be calculated

int index,SIZE;

unsigned long checksum_address=0x7F400;  //Destination address from calc_crc.crc file

unsigned short checksum_value;

checksum_value=*(unsigned short *)checksum_address;

SIZE=(sizeof(Memory)/sizeof(long));

for (index=0; index<SIZE; index++)

{

data = *(unsigned char *)Memory[index];

checked_data = crc16calc(checked_data, data);  // Calculate Checksum for each byte in
current memory  block

}

return (checked_data != checksum_value); // is calculated value equal to stored value
}

/*function to calculate checksum */

unsigned int crc16calc(unsigned short crc16, unsigned char next)

{

unsigned char  token1, token2;

unsigned short quick1, quick2;

token1 = (unsigned char)((crc16 >> 8) ^ next);

token2= (unsigned char)crc16;

crc16= (unsigned short)(token2 << 8);

quick1 = (unsigned short)(token1 ^ (token1 >> 4));

crc16 ^= quick1;

quick2 = quick1 << 4;

crc16 ^= quick2 << 1;

crc16 = crc16 ^ quick2 << 8;

return crc16;

}

int main(void)

{

if (CalcCRC16())

    printf("CRC fail!");

        else

          printf("CRC pass");

return 0;

}
```

Chapter 19
How to...

This chapter consists of the following topics.

- Switch Between Decoupled Parallel and Lock-Step Modes

19.1 Switch Between Decoupled Parallel and Lock-Step Modes

The following topic lists the steps to switch between the **Decoupled Parallel Mode** and **Lock-Step Mode (LSM)** . However, before proceeding ensure that you known the processor mode before proceeding with the actual mode.

- Things to Remember
- Switching from DPM to LSM using VLE

19.1.1 Things to Remember

Before Switching from DPM to LSM using VLE remember the following prerequisites.

- Changing the mode is done via target task.
- Each mode change target task is named as: `{processor}_[LSM|DPM]_[BOOKE|VLE].xml, LSM = lock step mode, DPM = decoupled parallel mode`
  - Select the FINAL processor mode. If you have a processor in LSM mode and you want to change it to DPM you will have to select the DPM task.
  - Depending on your run configuration select either VLE or BOOKE task. For each device that supports LS/DP there are two available script versions for VLE and BOOKE.
- Each target task requires a run configuration. The run configuration can be:

- Active debug context meaning the current active connection - if you already connected to the device.
- Manually set launch configuration. After importing the target task right-click to open the context menu and select **Change Run Configuration** .
- Dialog box prompting to select a launch configuration.
- Ensure that both the target task script and and the initialization tcl script from launch configuration have the same code encoding: VLE or BOOKE. However, it does not matter if the script is labeled for RAM or FLASH. To check the label, you can open the RSE system (aka processor) associated with that launch configuration and check the **Initialization** tab.

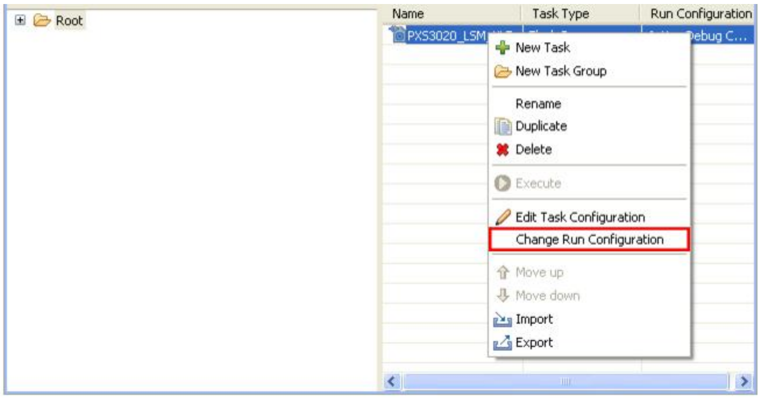


Figure 19-1. Select Change Run Configuration

19.1.2 Switching from DPM to LSM using VLE

To select the target task script by using the right the code encoding, VLE or BOOKE and the final processor mode, LSM or DPM, perform these steps.

- Import Target Task
- Execute Target Task
- Hardware Reset
- Unprotect Action
- Configure the Build Toolbar

19.1.2.1 Import Target Task

NOTE

> This procedure assumes that the Target Tasks view is visible in the perspective. If it is not visible, perform the steps in the Working with Flash Programmer topic to open the Target Tasks view.

1. Go to the Target Tasks view in either the C/C++ or Debug perspective.
2. Right-click on this view and select Import. Alternatively, click the icon on the **Target Tasks** view toolbar.

   The Open dialog box appears.

3. Navigate to <CWInstallDir>/MCU/bin/plugins/support/TargetTask/Flash_Programmer/E200 and select the XML file for the board's microcontroller. For example, select `MPC5643L_DPM_VLE.xml`.
4. Click Open.

   The `MPC5643L_DPM_VLE` task appears in the Target Tasks view.

5. Double-click on the `MPC5643L_DPM_VLE` task to examine its contents.

   The <target> Flash Programmer Task editor window appears, and displays the memory settings and actions for the task. Notice the actions to erase, program, and verify the contents of flash memory in the Flash Programmer Actions group. These actions execute in the order as they are displayed in the table, from top to bottom.

6. Execute the target task. For more information, refer Execute Target Task.

19.1.2.2 Execute Target Task

To execute the task:

1. Right-click on the imported task.
2. Select Execute. Alternatively, select the imported task and click the Execute icon on the Target Tasks view toolbar.
3. CodeWarrior establishes contact with the board, erases the microcontroller's flash memory, downloads the code, and verifies that the contents of flash match those of the file. In this case the target task will erase the shadow flash which means that it will set the Lock step bit.
4. Reset the board. For more information, refer Hardware Reset. This step is mandatory in order to use the new LS configuration.

NOTE

When changing the device back to DPM the shadow flash will be programmed. However, it is possible that the shadow flash sector is protected. In case the shadow flash sector is protected then the target task will fail and you will need to add an unprotect action to target task. This action need to be run before any other actions.

19.1.2.3 Hardware Reset

To reset the hardware, perform these steps.

1. Unplug the power cord.
2. Plug the power cord again.

   or

   Simply press the reset button on the hardware or target board.

19.1.2.4 Unprotect Action

To add an unprotect action to the target task, perform these steps.

1. Select the Protect/Unprotect action from the **Add Action** drop-down list.

   The Add Protect/Unprotect Action dialog box appears.

2. Check the All Device check box to add an unprotect action on the full device.
3. Click the Add Unprotect Action button.
4. Click Done.

   The Add Protect/Unprotect Action dialog box closes and the added protect or unprotect actions appear in the Flash Programmer Actions table in the Flash Programmer Task editor window.

NOTE

None of the target task restores the shadow flash user registers. If the device has some user defined configuration in shadow flash then executing the changed target task will erase the configuration also.

19.1.2.5 Configure the Build Toolbar

The Build toolbar feature provides an easy way of managing the active configuration of projects grouped in a working set. The toolbar is made of two elements:

- **Projects drop-down list**: Enables you to select the active configuration.
- **Hammer icon**: Enables you to start the build process of the selected configuration.

NOTE

The default value is **(Active)** and its means that all the projects are in the workspace.

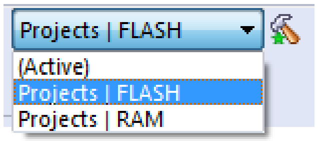
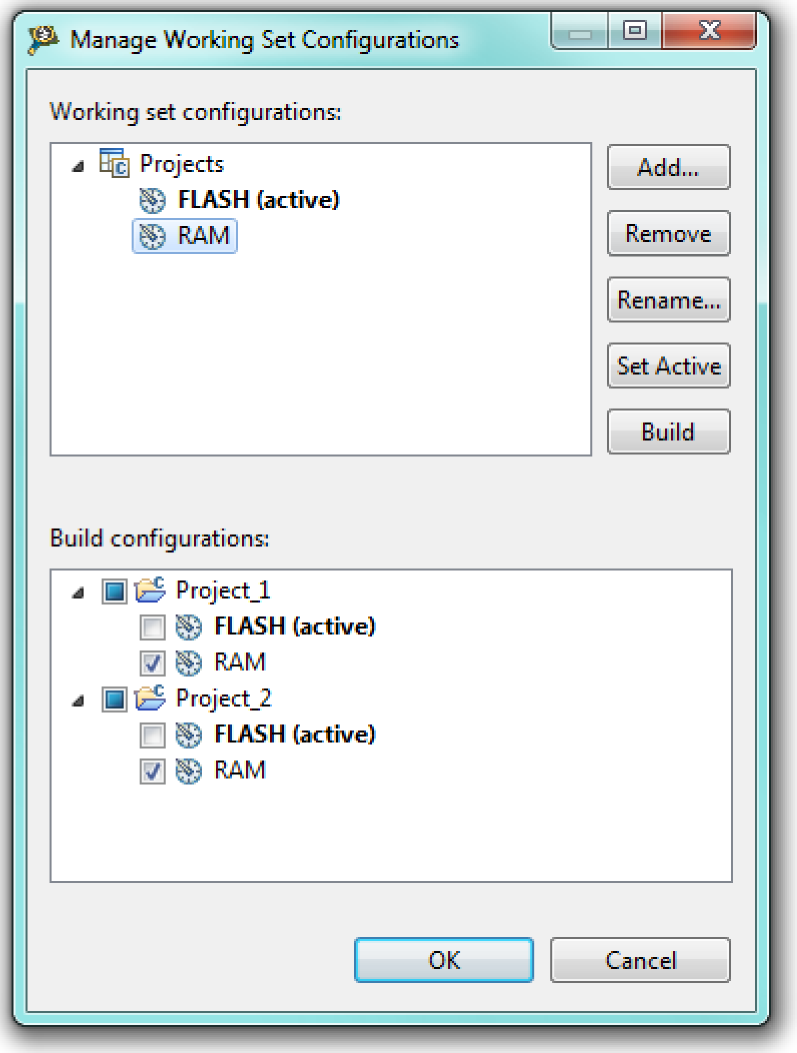


Figure 19-2. Build Toolbar

To create some entries in the drop-down list, perform the following steps:

1. Create a C/C++ working set with one or more projects. This can be done in the dialog, which is opened from the “Project/Build Working Set/Select Working Set…” menu, by clicking the “New…” button and then selecting the “C/C++“ type.



Working set configurations:

- Projects
  - **FLASH (active)**
  - RAM

Build configurations:

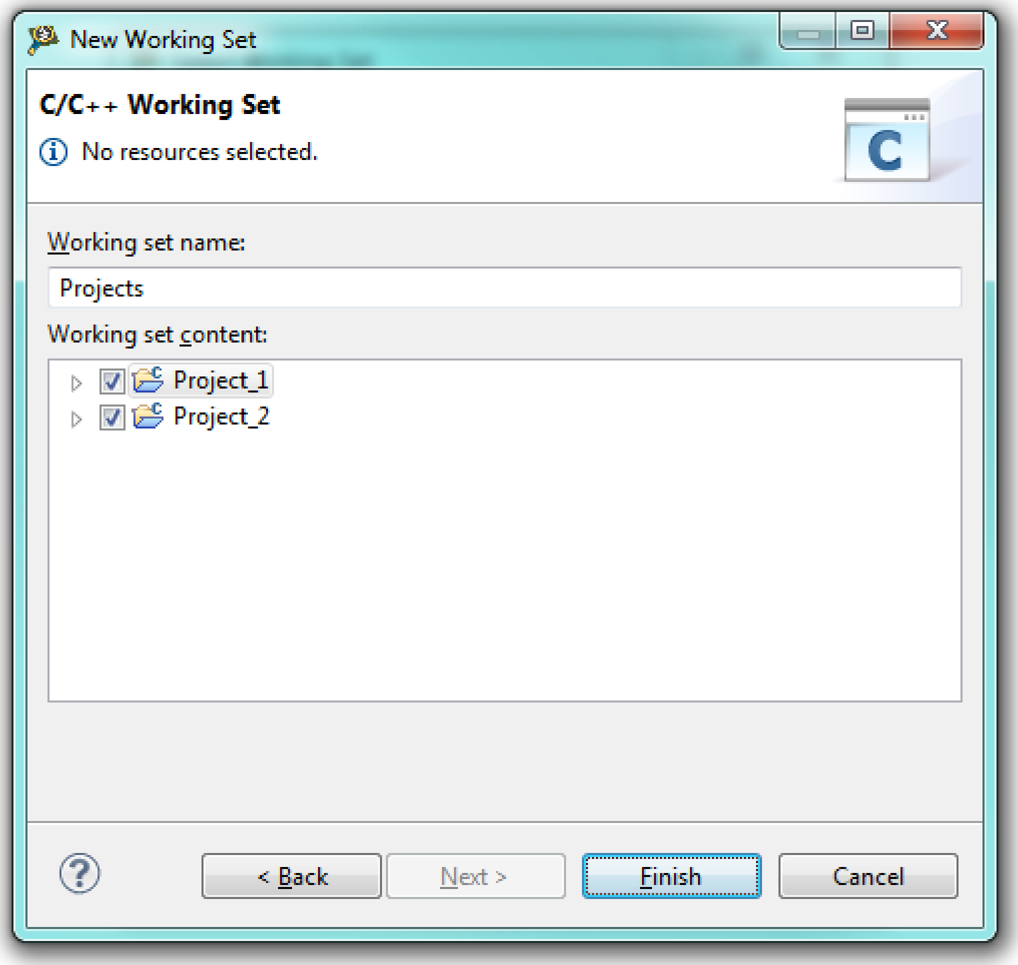
- ■ Project_1
  - □ **FLASH (active)**
  - ☑ RAM
- ■ Project_2
  - □ **FLASH (active)**
  - ☑ RAM

Figure 19-3. Build Toolbar

2. Create a working set configuration(s). To do so, go to the “Project/Build Configurations/Manage Working Sets…” menu and add one or more configurations to the working set created in step 1.

NOTE

If you do not plan to use this feature, you may hide the Build Toolbar in the “Customize Perspective” dialog opened from the “Window/Customize Perspective…” menu.



New Working Set

C/C++ Working Set

No resources selected.

Working set name: Projects

Working set content:

☑ Project_1

☑ Project_2

Figure 19-4. Build Toolbar

Chapter 20
S12Z IEEE-754 Floating Point Library

This chapter presents an implementation of the floating point arithmetic as described in the IEEE-754 standard. The following floating point routines for the S12Z device family are implemented. Refer the standard for detailed description of their functionality

- Basic floating point operations for single precision and double precision: addition, subtraction, multiplication, division
- Conversion to and from integer (16-bit, 32-bit and 64-bit) and floating point format (32-bit and 64-bit).
- Conversion between 32-bit and 64-bit floating point
- Comparison functions
- ANSI functions: `frexp/f, ldexp/f, modf/f` for single and double precision
- Functions for manipulating exception flags: `saveAllFlags`, `restoreFlags, testFlags`, `testSavedFlags`, `lowerFlags`, and `raiseFlags`.

Floating point functions are provided in the form of libraries and source code, both C and assembly. The implementation is ready for use with the CodeWarrior compiler.

The implementation demonstrates a good balance between functionality and performance and for this reason does not strictly follow the floating point standard. In particular, the implementation provides several library variants, each of them differing in compliance level to the standard.

Table 20-1. ansif and ansid Library Features

Features	ansif	ansifc	ansid	ansidc
64-bit double	No	No	Yes	Yes
Rounding	None	Round to the nearest even	None	Round to the nearest even
Non-numerical values	Yes	Yes	Yes	Yes
Sub-normal values	Yes	Yes	Yes	Yes
Exceptions	No	Yes	No	Yes
ANSI functions	Yes	Yes	Yes	Yes

1. `ansif.lib`: This library is the smallest and generally the fastest. The double type is mapped to 32-bit. Apart from basic functionality (arithmetic operations, conversions), this version can handle denormalized numbers, as well as non-numerical values.
2. `ansidr.lib` / `ansidm.lib*`: This library provides the best compromise between functionality and performance. It is a bare bone implementation of the standard with double precision capabilities. Denormalized numbers and non-numerical values are supported.
3. `ansidr.lib` / `ansidm.lib`: This library offers a good level of IEEE-754 standard compliance, without sacrificing much on performance. This version supports rounding, namely the "round to nearest even number" rounding mode. Additionally, it offers support for exception flags (overflow, underflow, inexact, invalid, divide by zero). Comparisons with NaNs (unordered comparisons) behave according to the standard.

NOTE

"r"-suffixed and "m"-suffixed libraries differ in terms of calling convention for 64-bit data types. Depending on the -bit64_code_gen option supplied to the compiler, the appropriate library must be linked: ansidr(c) for a register-based calling convention and ansidm(c) for a stack-based calling convention.

NOTE

A detailed discussion regarding use of the different floating point features imposed by the IEEE-754 standard is beyond the scope of this chapter and shall not be provided. However, users are reminded that this subject is non-trivial. It is recommended that users familiarize themselves with the appropriate literature in order to use such features correctly.

20.1 Usage

The floating point libraries should be used by adding the respective library to a CodeWarrior project. The CodeWarrior linker will link the project compiled binaries against the added library.

All functions have been designed to execute as fast as possible in the presence of normalized numbers as input arguments. In the case where sub-normal numbers are supplied, the execution time may be longer. In any case, it should be noted that a frequent appearance of sub-normal numbers in floating point computation may indicate that an implemented algorithm needs some refinement.

20.2 Supported IEEE-754 Features Description

The description includes:

- Format
- Non-numerical Values
- Sub-normal Values
- Unordered Comparisons
- Rounding
- Exception Flags

20.2.1 Format

The implementation supports both the single precision (32-bit) and double precision (64 bit) formats described in the IEEE-754 standard. The ansif library only supports single precision, because double precision numbers are restricted to 32-bits for performance purposes. 80-bit extended precision is not supported.

20.2.2 Non-numerical Values

All library variants support non-numerical values (NaN, Infinity). The compliant libraries support quiet and signaling NaNs and handle comparisons with NaN according to the standard.

20.2.3 Sub-normal Values

Sub-normal values are supported by all library variants. It is not possible to treat the sub-normal values in a different way (for example, as zero - "flushing to zero").

20.2.4 Unordered Comparisons

Floating point comparisons are mapped to C operators: ==, <, >=, etc. In the case of non-compliant libraries, all comparisons are considered ordered, which means comparisons with NaN have undefined results. Compliant libraries support unordered comparisons, which have the added functionality of returning false for every comparison which involves a NaN, except "not equal". In order to enable unordered comparisons, the "fp_compliant" option needs to be passed to the compiler. The following table better illustrates unordered comparisons:

Table 20-2. Unordered Comparison

		==	!=	>	<	>=	<=
f	NaN	False	True	False	False	False	False
NaN	f	False	True	False	False	False	False
NaN	NaN	False	True	False	False	False	False

20.2.5 Rounding

The `ansif` and `ansid` libraries use no rounding. Note that no rounding might yield different results compared to truncation. The compliant libraries use only one rounding mode, namely the "round to nearest even number". Although there is only one rounding mode, it provides the best precision and is the standard choice for the vast majority of use cases. With the exception of implicit floating point to integer conversions, all functions are correctly rounded.

When no rounding mode is used, precision loss is generally 1ulp. However, in the case of subnormal numbers, the precision loss may be greater.

20.2.6 Exception Flags

Compliant libraries offer support for exception flags, according to the IEEE754 standard. For each exception, the implementation provides a status flag that is set when the exception occurs. The flag is cleared only at the user's request. The user can test and alter flags individually, or several at a time. Furthermore, all flags can be saved and restored.

There are five supported exceptions:

1. invalid operation: Signaled if an operand is invalid for the operation. For example: Inf - Inf, 0 * Inf, 0 / 0, Inf / Inf and others. The result will be a NaN.

2. division by zero: Signaled if the divisor is 0 and the dividend is a finite non-zero. The result will be a correctly signed Inf.
3. overflow: Signaled when the destination format's largest finite number is exceeded by the rounded result. The result will be a +Inf.
4. underflow: Signaled when a tiny (between +/- 2Emin) non-zero result is created. The result will be 0.
5. inexact: Signaled when the result of an operation is not exact, including the case of overflow.

The five exception flags are defined in except.h: `FP_EINVALID`, `FP_EDIVZERO`, `FP_EOVERFLOW`, `FP_EUNDERFLOW`, `FP_EINEXACT`.

For example, in order to check if an operation has overflowed, the testFlags function is used:

```
if (testFlags(FP_EOVERFLOW)) { do_something(); }
```

Multiple flags can be checked at the same time:

```
if (testFlags(FP_EOVERFLOW | FP_EUNDERFLOW)) { do_something(); }
```

Flags that have been set by a certain floating point operation are not cleared by subsequent operations. The only way to clear flags is by using the `lower_flags` function:

```
lowerFlags(FP_INVALID | FP_EOVERFLOW | FP_EUNDERFLOW)
```

Finally, flags can be saved using the `saveAllFlags()` function. Any number and combination of flags can be restored to their previous state using the `restoreFlags(char, char)` function.

20.3 Performance

This topic discussed the code size and the stack consumption.

20.3.1 Code Size

The following table lists the library and the code size in bytes.

Table 20-3. Library and Code Size

Library	Code Size (bytes)
ansif	1600
ansifc	2200

Table continues on the next page...

Table 20-3. Library and Code Size (continued)

Library	Code Size (bytes)
ansid	2500
ansidc	3200

The following table list the operation and compliance.

Table 20-4. Operation and Complaince

Operation	Default	Compliant
Float addition	300	430
Float multiplication	450	590
Float division	370	530
Float to long conversion	170	230
Double addition	360	550
Double multiplication	650	800
Double division	550	730
Double to long long conversion	280	300

20.3.2 Stack Consumption

The difference in stack consumption between compliant and non-compliant libs is negligible. Maximum stack consumption for float libs is 50 bytes; maximum stack consumption for double libs is 90 bytes.

Note that these represent maximum values for the runtime library. Standard library functions like sin, cos, etc. may induce serious stack overhead. Furthermore, operation chains like a + b * c create temporaries which take up additional stack space.

Table 20-5. Stack Consumption

Operation	Stack Consumption
Float addition	32
Float multiplication	50
Float division	36
Float to long conversion	24
Double addition	62
Double multiplication	90
Double division	56
Double to long long conversion	30

Index

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number CWMCUDBGUG
Revision 10.6, 03/2014

freescale™